Second ARRL Amateur Radio Computer Networking Conference

March 19, 1983

San Francisco, California

Hosted by:

Amateur Radio Research and Development Corp.

Pacific Packet Radio Society

Second ARRL Amateur Radio Computer Networking Conference

Coordinators:

Hank S. Magnuski, KA6M

Paul L. Rinaldo, W4RI

Committee:

Stu Neblett, K6VCO

Bob Reiling, W6JHJ

Curtis Spangler, N6ECT

San Francisco Radio Club

American Radio Relay League

Newington, CT USA 06111

Contents

THE PACSAT PROJECT

Den Connors, KD2S
PACSAT Project Manager
Radio Amateur Satellite Corporation - AMSAT

AMSAT has begun the design and development of a new form of Amateur Satellite. The PACSAT series of satellite systems has as a design goal total global access by all hams to a store-and-forward packet radio message handler.

Introduction

AMSAT is proposing the design and prototyping of a satellite-based experiment for advanced digital packet satellite communications experiments. This system, called PACSAT, will use internationally-allocated Amateur Radio Service frequencies. The PACSAT system will connect a grid of ground-based amateur radio local area networks in the United States and many other countries via a common store-and-forward packet repeater operating in the Amateur Satellite Service.

This paper details the reasons behind such a satellite. Following the design concepts, a description of the entire system is given, and a list of technical parameters for each of the defined subsystems is shown. The current outline of tasks and scheduling follows, with a description of the efforts of groups already engaged in the initial design effort.

The PACSAT Concept

The Amateur Radio communities in the United States and other countries are currently experimenting with digital networks on radio channels. These networks are using techniques already in place on the national telecommunication networks known collectively as packet networking.

Packet radio systems have a set of benefits unusual in present amateur radio systems. Large numbers of stations may share a common frequency, and use multiple access packet techniques to multiplex several sets of users in the time domain; very high spectrum utilization is accomplished by keeping all of these users on the same channel. A second benefit of the single shared channel is the ability to find all other users of the packet radio system. No searching of a wide band of frequencies is required; connectivity is maximized. The need of multiple access techniques to detect successful transmissions yields a third benefit, that of reliable transmissions. Any message that arrives at destination has had its data integrity checked. This inherent reliability may well open a series of possibilities for improving emergency traffic handling, one of amateur radio's most important aspects.

As experiments continue on ground-based packet radio local area networks, a new class of satellite is being considered to handle linking of both individual ground stations and local area networks. The PACket radio SATellite (PACSAT) system is designed to provide a store-and-forward digital repeater which is available to all groups around the world for fully global network coverage. The satellite provides this coverage by occupying a low-earth orbit (LEO), which has several benefits. The close proximity of passage, relative to geo-synchronous satellite, allows easy access, with good "link margins". There are thousands of amateur radio earth stations that are already configured to operate on this class of satellite. Additionally, proper choice of orbital parameters allows a sun-synchronous orbit, where passage of the satellite occurs at the same times each day, providing an easy means of scheduling transmissions. This orbit then provides both 100% global coverage and very fair access, and creates a powerful new use of a well-known class of amateur satellite.

There are several purposes for providing such a system in the Amateur Satellite Service. PACSAT will provide a wide-availability vehicle for advanced experimentation, and a prototype system for a new class of satellite service involving reliable transmission of data to remote sites and isolated users, regardless of location. Several internationally-based organizations have expressed interest in just such capabilitiies, and this gives AMSAT the oppurtunity of spear-heading a potentially major new push in low-cost satellite systems, much in consonance with the F.C.C. charter for the Amateur Radio Service as a "proving-grounds" for new technology. The Volunteers in Technical Assistance (VITA), a non-profit firm dedicated to advancing the level of technilogical expertise of less-developed countries, is actively pursuing the coordination of such vanguard activities, and is working directly with AMSAT on the PACSAT development.

Other benefits result from the use of digital techniques. Considerable improvements may be made to emergency communications, as reliable, high availability links compatible with global mobile and portable radio service requirements are provided. Additionally, a spin-off benefit for AMSAT itself is the attracting of the new, computer-aware members of the Amateur community into the Amateur Satellite Service.

A possible third set of benefits may be spun off the PACSAT system indirectly; the opportunity exists for designing new types of low-cost satellite launching and propulsion systems as a part of this next generation of Amateur spacecraft.

Such a system would provide a number of functions. In addition to the primary use as a world-wide store-and-forward link, or "flying mailbox", the PACSAT experiment could provide real-time regional linking (standard LEO amateur mode). As mentioned, both local network concentrators (gateways) and individual users could access the satellite. Finally, the system would provide the mechanism for advanced testing of network systems concepts, hardware, software and protocols to be used by packet radio networks in the future.

Other benefits result from the use of digital techniques. Considerable improvements may be made to emergency communications, as reliable, high availability links compatible with global mobile and portable radio service requirements are provided. Additionally, a spin-off benefit for AMSAT itself is the attracting of the new, computer-aware members of the Amateur community into the Amateur Satellite Service.

A possible third set of benefits may be spun off the PACSAT system indirectly; the opportunity exists for designing new types of low-cost satellite launching and propulsion systems as a part of this next generation of Amateur spacecraft.

Such a system would provide a number of functions. In addition to the primary use as a world-wide store-and-forward link, or "flying mailbox", the PACSAT experiment could provide real-time regional linking (standard LEO amateur mode). As mentioned, both local network concentrators (gateways) and individual users could access the satellite. Finally, the system would provide the mechanism for advanced testing of network systems concepts, hardware, software and protocols to be used by packet radio networks in the future.

PACSAT System Description

PACSAT is an extraordinarily complicated system, rather similar in complexity to the Phase III spacecrafts. In addition to all of the required satellite support subsystems on board the spacecraft, there are two experimental packages, each consisting of multiple uplink channels, common downlink channels, and modems, coder/decoder link-access devices and control microprocessors with interfaces to a common satellite message processing unit (system control unit, or SCU). As if that isn't bad enough, the rigid packet environment demands structured ground stations with all of the familiar hardware (less the directional antennas, as we shall see), and perhaps several micropressors for handling both the data stream and the automatic station control functions. The age of the microprocessor is upon Amateur Radio.

To ease the burden of trying to understand the whole system, PACSAT can be broken up into subsections, each with well-defined interfaces to other sections. A description of each section or interface follows. Please note that, although the conceptual design has been finished, many design groups are hard at work coming up with the specifications for their parts of the overall system, so that nothing below can be consrued to be the "final word".

Spacecraft

As mentioned, the orbit of PACSAT would be sun-synchronous, that is, appearing at the same time each day. UoSAT/OSCAR 9 has this type of orbit, and displays this property. Additionally, such an orbit guarantees at least two passes per day will be seen by ALL corners of the Earth.

The PACSAT satellite system may be broken into the spacecraft itself, and the experimental packages. The interfaces are defined to be spacecraft/ experiment and spacecraft/ground station.

Two options are available for placing PACSAT experiments into space. The possibility of riding the packages inside of a spacecraft built primarilly for other purposes exists, and allows the PACSAT design team to avoid the additional complexity of designing and building all of the required subsystems. AMSAT has looked in particular at the future launch opportunities available on the Conestoga-series of launch vehicles to be provided as a commercial venture by Space Systems of America, Incorporated. SSI will be launching payloads directly into low-earth orbit, providing a mechanism for direct injection of PACSAT into its final orbit without requiring on-board propulsion systems in the satellite.

A second opportunity is more in line with AMSAT's traditional method of designing the satellite "from the ground up", and will likely provide many more opportunities for future launches. The Space Transportation System (Space Shuttle) has the option of carrying into space sets of three "Get-Away Special" canisters, or GAS cans. Although these cans have traditionally been reserved for inexpensive access for experimenters who did not require throwing their experiments into space, recent discussions with NASA have shown promise for using such a can as a launch opportunity. A satellite would be placed inside the can, with a mechanism in place to allow the Shuttle crew to remotely open the lid and push the spacecraft into the void (hopefully after opening the Shuttle bay doors).

This new opportunity has two tremendously useful aspects: GAS can opportunities are CHEAP ($10,000) and potentially plentiful. The drawbacks are the requirements of building such spacecraft as would be required to fit into the can, and providing a propulsion mechanism for altering the very low orbit into whicn the Shuttle would place the unit, so that a final, more stable orbit would be available. As it happens, there are active international AMSAT groups that are very excited with the possibility of providing both spacecraft and propulsion.

The University of Surrey spacecraft design team (UoSAT) has expressed an interest in continuing their advanced low-cost spacecraft design and construction projects, and view PACSAT as an excellent opportunity for using their integration expertise, and for providing a vehicle to carry other experiments of interest to their group.

The AMSAT/DL team at the University at Marburg, West Germany, has been discussing the possibility of providing an innovative spacecraft engine which would be ideal for such a craft as PACSAT - a steam engine, not unlike those first designed by Hero in ancient Greece. The mechanism for generating steam in space is not difficult, and inpinging sunlight on external water tanks could provide a large part of the energy required to heat the water. Heating coils electrically powered in the area of the super-heated steam nozzles would finish the heating job. Although this concept seems a little far-fetched at first, calculations prove the amount of water required to alter the orbit of PACSAT is quite modest. Further, the ever-present problem of safety to the Shuttle crew is very much reduced by having spacecraft with extremely non-volatile fuels such as water! Gradual pushes from the steam nozzles at opposites sides of the orbit will nudge PACSAT into its final orbit, and residual water could be used to further occasionally alter the orbit to keep it in a sun-synchronous plane.

PACSAT Communications Experiment Package

Each of two packages will contain a set of uplink and downlink channels with associated analog and digital hardware. Current designs are targetted for typically four uplink channels, each dynamically configurable with respect to data rate. One high-speed downlink channel will be used to support the uplinks, and to provide control over the smart ground stations. For an excellent review of the design effort for the modulation techniques and access modes of these channels, see the paper "Modulation and Access Techniques for PACSAT" by Phil Karn, which is included in these proceedings.

Supporting these communications channels will be a series of filters, oscillators and amplifiers, along with microprocssors and buffer memory for channel control and support of link access protocols. These processors, with perhaps one or two channels per processor, will allow the demodulatiors chosen to be both adaptive in data rate and frequency agile.

The set of packages will have a common system controller and main memory unit (RAMUNIT). The software to support the higher-level protocols and application programs to be resident in the SCU will be loadable from the ground, a technique now common in the Amateur Satellite Service. A memory package in the megabyte range is being investigated.

Spacecraft/Experiment Interface

The spacecraft will provide the envvironment for PACSAT, including power, antennas and shielding from the extremes of space. A separate processor will handle the spacecraft's housekeeping functions, and separate communications channels will be available for satellite command. Standard interfaces will define nd stations will be fairly
complicated, requiring smart controllers to handle the requirements of frenquency agility in the transmitters, and of linking, networking and presentation control.

To allow users to ease into packet radio satellites, a gradual upgrade path is to be provided for PACSAT use. A required piece of equipment will be the modem, which will include a modulator, demodulator and pass-through path for transmitter push-to-talk and frequency control. This modem will be capable of operating as a stand-alone modem attached to one of the current types of packet radio terminal node controllers (TNC).

Operation of the TNC and modem pair with a standard set of 440-MHz transmitter and 2-meter receiver will allow operation at 1200 bauds. Higher speed operation will require a separate rf deck, with direct access to IF strips. Speeds of up to 9600 bauds are planned.

A final touch would be a custom TNC, specifically designed for this system, and allowing direct interface to other TNCs for ground-based internetwork linking.

It should also be noted that conservative link margin calculations have shown that, with modest transmitter power on board the spacecraft and standard powere available to ground stations (around 25 watts), the requirement for having directional antennas is not necessary. Simple gain verticals like 5/8th whips on 440 and 2 will probably be quite adaquate, especially at lower baud rates like 1200 bauds.

Spacecraft/Ground Station Interface

The PACSAT Project intends to use omnidirectional antennas on two of the most popular vhf/uhf bands, in a mode which will be familiar to Phase III users. Uplinks will be available at aroung 435 MHz, and choice of the proper channel will be made by the ground station controller, following the command requests of the satellite. The common downlink will appear at the edge of the Amateur Satellite allocations, probably around 145.806 MHz for one package and 145.994 MHz for the other.

The modulation technique, synchronization requirements, encoding mode and related parameters are to be determined, based on experiments to be performed by two different design teams this spring. It is assumed that either differentially-encoded phase-shift keying or minimum shift keying at rates in the 1200 to 9600 baud range, perhaps adaptively availble, are the most likely candidates.

The link-level access protocols, that is, the addressing and error detection schemes are planned to be compatible with the AX.25 Amateur packet radio protocol standard. This protocol has already been implemented by several groups, and is a de facto AMSAT standard for all currently-planned packet satellite efforts.

The network protocol will probably support AX.25 network-level protocol, and perhaps also less complicated (and less reliable) "datagram-type" protocols as well.

The memory interface between ground stations and the on board RAMUNIT will be little more than a virtual disk drive. with a very noisy connecting link. On top of this protocol will lie an applications program which will provide a number of message and file services. Experimenters will be provided with lower-level accesses to the system where such access does not significantly disrupt normal use of the system.

PACSAT Project Status

The final conceptual review meeting was held in February 1983, and several of the design groups attended, including representatives from both VITA and the University of Surrey. Many of the more sophisticated concepts were thrown away to provide an easier target for scheduling. There will be a set of subsystem design meetings at this conference, and further meetings to be held later this spring. Negotiations are currently underway with the candidate launch agencies and design support groups.

System design is likely to be completed early this summer, with deliverable items to be integrated and tested this fall. Following critical design review meetings, spacecraft-ready subsystems will be prepared and shipped to the integrating agancy by next spring. Such a schedule would allow AMSAT to take advantage of possible launch opportunities as early as late 1984. Slippages will be inevitable however, and more realistic times will in general coincide with the more likely target launch dates, early 1985 to 1986.

The project now has the support of twelve differennt design groups from four different countries, but is still in need of qualified hardware and software designers to help review all aspects of the current design, and provide needed manpower with several of the more important subsystems. PACSAT is an all-volunteer effort, and will require careful evaluation by the general user community during its initial phases to confirm design parameters and provide guidance in the utility of the various modes. It is hoped that this system will not only provide many services which are forecast for the digital fututre of ham radio, but also create a whole new set of users and uses yet to be imagined.

AX.25 LEVEL 2 PROTOCOL

Terry Fox, WB4JFI
Vice-President, AMRAD
1819 Anderson Road
Falls Church, VA 22043

Abstract

This paper contains the latest draft of the AX.25 protocol specification. This is the first public release of this draft. Earlier drafts have been given to specific inviduals for comment and as a reference for software development. Changes should be expected. Please check the AMRAD Newsletter for announcements of later versions.

History

Over the years there have been several protocols suggested for use at layer 2 of the ISO Open System Interface Reference Model (OSI-RM) over Amateur Radio. The one system that has been in use is based on the IBM SDLC protocol, and it has been working as far as it went. One of the immediate problems that came up with SDLC was that its address field of SDLC is very limited (being one byte long), causing problems if there are many amateurs on at a time.

Trying to come up with a protocol that everyone would agree to seemed like an almost impossible task a year ago. What we at AMRAD decided to do was to go over the various protocols in use or available to the amateur, figure out the best and worst parts of each protocol and see if the protocol could be "enhanced" to work properly over the amateur radio enviroment. After reviewing the various protocols around and talking with people in the computer networking industry, we decided to push the X.25 standard, modified to allow a larger address field. At about this time, a group of amateurs in New Jersey were coming to the same conclusion, so about mid-June of 1982 the two groups got together and after two weekends came to an "understanding" on a level 2 protocol. The most delicate part of the negotiations between the two groups concerned the name to be given this protocol. In order to not step on anyones toes, it was decided to call the protocol AX.25, which stands for Amateur X.25.

The next step in the evolvement of AX.25 was that in October of 1982, AMSAT hosted a gathering of some of the leaders in amateur packet radio. AMRAD was at the meeting, along with representatives from TAPR, SLAPR, AMSAT, and PPRS. Three days of intense discussion followed, and an agreement was finally reached on a nation-wide compatible protocol. AX.25 was then modified to be compatible with this new protocol (basically the only major changes were an additional extension of the address field, and the addition of a Protocol IDentifier, or PID field).

The rest of this paper will describe the basics of the AX.25 level 2 protocol.

AX.25 Layer 2 Protocol Specifcation

This protocol conforms with the ISO Recommendations 3309, 4335 (including DAD 1&2) and 6256 high-level data link control (HDLC) and uses some terminology found in these documents.

This protocol also conforms with ANSI X3.66, describing ADCCP, balanced mode.

This protocol is written to work equally well in either half- or full-duplex amateur radio enviroments.

This protocol has been written to work equally well for either point-to-point connections, or connections made thru a larger device, such as a metropolitan network controller (MNC).

This protocol does allow the establishment of more than one layer 2 (link layer) connection per device, if the device is so capable.

This protocol also follows in principle the CCITT X.25 recommendation, with the exception of an extended address field and the addition of the Unnumbered Information (UI) frame.

Most layer 2 protocols assume that one large device (generally called a DCE, or data circuit-terminating equipment) is connected to several smaller devices (usually called a DTE, or data terminating equipment). AX.25 assumes that both ends of the link are balanced, thereby eliminating the two different classes of device.

Frame Structure

Level 2 packet-radio transmissions are sent in small blocks, called frames. These frames are made up of smaller parts, called fields. Fig. 1 shows how the three types of frames are made up. Fig. 1 shows the frames in the same bit order that most packet articles show them. Unfortunately, this method has led to some confusion, since the least-significant bit (LSB) is to the left rather than to the right, as most people would ordinarily assume. I am pointing this out early in this paper to prevent mass confusion as I progress. Later on, I will switch to a hopefully more understandable way of showing the frame ans its components.

Field Definitions

The frame is made up of several parts, called fields. Each of these fields is made up of an integral number of octets (or bytes), and serves a specific function.

Flag Field

Since amateur packet radio is a bit-oriented protocol, the only way to tell when one frame is over and another is starting for sure is to delimit each frame with a certain bit sequence both at the beginning and the end. This is the job of the flag field. A flag consists of a zero followed by six ones followed by another zero, or 01111110 (7E hex). Due to the bit stuffing mentioned above, the only time this sequence is allowed is at the beginning and end of a legitimate frame.

Address Field

The address field is used to identify both where the frame came from and what the destination of it is. In the CCITT recommendation X.25, this field is only one octet long. This permits at most 256 users per level 2 channel, and since some bits of this field were used for other purposes, the real number of users were about thirty per level 2 channel. Both the HDLC and ADCCP recommendations allowed the address field to be extended, so we decided to extend the address field per their recommendations in the amateur version of X.25 to include the callsigns of both the destination and source amateur radio stations.

The method used to extend the address field will be described shortly.

Control Field

The control field is used to identify the type of frame and control several attributes of the level 2 connection. It is one octet in length, and its encoding will be discussed in a following section.

PID-Field

The Protocol Identifier (PID) field is used only in information frames, and identifies what kind of layer 3 protocol, if any, is in use. Its encoding is as follows:

M S B L S B	
xx00xxxx	Reserved at the moment.
xx01yyyy	AX.25 layer 3 implemented.
xx10yyyy	AX.25 layer 3 implemented.
11110000	No layer 3 implemented.
11111111	Escape character. Next byte contains more PID information.

Where:

1. An x indicates a "don't care" bit.
2. A y indicates all combinations used.

Information Field

The information field is used to convey the actual user data from one end of the link to the other. I fields are allowed in only three tyes of frames, the I frame, the UI frame, and the FRMR frame. The I field can be up to 256 octets long, and should be an even multiple of octets long. Any information in the I field should be passed along the link totally transparently, except for any zero-bit insertion necessary to prevent flags from accidentally appearing in the I field.

Frame Check Sequence

The frame-check sequence is a sixteen-bit number calculated by both the sender and receiver of a frame. It is used to make sure that the frame was not corrupted by the medium used to get the frame from the sender to the receiver. It is calculated in accordance with ISO 3309 (HDLC) recommendations.

Bit Stuffing

In order to assure that the flag sequence mentioned above doesn't accidentially appear anywhere else in a frame, as the frame is being sent it should be monitored, and if more than five contiguous ones are detected, a zero bit should be added between the fifth and sixth ones, eliminating the possibility of a flag appearing in the frame other than where it belongs. The receiver of five ones, a zero, and more ones should automatically eliminate the inserted zero before passing the data on.

Bit Order of Transmission

With the exception of the FCS field, all other fields in an AX.25 frame should be sent starting with the least-significant bit. In accordance with HDLC practices, the FCS should be sent most-significant bit first.

Frame Abort

If a frame must be prematurely aborted, at least fifteen contiguous ones should be sent with no bit stuffing added.

Invalid Frames

Any frame consisting of less than 136 bits, or not bounded by opening and closing flags, or not octet aligned (an integral number of octets) should be considered an invalid frame by the link layer.

Address Field Encoding

The address field of all frames should be encoded with both the destination and source amateur callsigns of the frame. If a level 2 amateur "repeater" is to be used, its callsign should also be in the address field. AX.25 follows the HDLC recommended method of extending the address field in order to fit all this information into the address field.

Basically, the way the HDLC address field is extended beyond one octet is to reserve the least-significant bit of each octet for what is called an "extender bit". This bit is set to zero if the next octet contains more address field information, and is set to one if this is the last octet. To make room for this extender bit, the amateur radio call sign information is shifted one bit to the left.

The actual encoding techniques for both non-repeater and repeater operation follows.

Non-Repeater Address-Field Encoding

If a level 2 repeater is not being used, the address field is encoded as shown in Fig. 2. The destination address is the call sign of the amateur radio station that the frame is addressed to, while the source address contains the amateur call sign who sent the frame. These call signs are the call signs of the two ends of a level 2 AX.25 link **only**, not of any other station, such as the destination of a packet going thru an intermediary link. Those addresses should be in a higher layer, not layer 2.

A1 thru A14 are the fourteen octets that make up the two address sub-fields of the address field. The destination sub-address is seven octets long (A1 thru A7), and is sent first. This will allow the receivers of the frame time to check the destination address sub-field and see if the frame is for them while the rest of the frame is being received. The source address sub-field is then sent in octets A8 thru A14. Both of these sub-fields are encoded in the same manner, except for the last octet having the HDLC address extender bit set. Since they are basically the same, only the destination sub-address will be outlined.

There is an extra octet at the end of each address sub-field that allows room for a Secondary-Station Identifier (SSID) and three additional bits for future expansion. The SSID field allows an Amateur Radio operator to have more than one packet radio station. This is useful when an amateur wants to put up a repeater in addition to his regular station for example.

Appendix A shows a typical AX.25 frame in the non-repeater mode.

Destination Sub-Field Encoding

Fig. 3 shows how an amateur call sign is placed in the destination address sub-field, occupying octets A1 thru A7.

Octet	ASCII	Bin.Data	Hex Data
A1	W	10101110	AE
A2	B	10000100	84
A3	4	01101000	68
A4	J	10010100	94
A5	F	10001100	8C
A6	I	10010010	92
A7	SSID	0RRSSID0	

Bit Position--> 76543210

Fig. 3. Destination Field Encoding

Where:

1. The top octet (A1) is the first octet sent (sort of like popping it off the top of the stack), with bit 0 of each octet being the first bit sent, and bit 7 being the last bit sent.

2. The first (low order or bit 0) bit of each octet is the HDLC extender bit, which is set to zero on all but the last octet in the address field, where it is set to one.

3. The bits marked "R" are reserved bits. they may be used in an agreed upon manner in individual networks. If they aren't implemented, they should be set to one.

4. The characters of the callsign should be standard seven-bit ASCII (upper case only) before being shifted left to make room for the extender bit. If the callsign is less than six characters long, it should be padded at the trailing end with ASCII spaces between the end of the callsign and the SSID octet.

5. The SSID portion of the last octet has been intentionally left vague at this point, and is left up to the individual station to assign. The only recommended restriction is to reserve the all-one condition (1111) for an all-call SSID in case one wants to reach an amateur but doesn't know what SSID that amateur operates under.

Level 2 Repeater Address-Field Encoding

If a frame is to go thru a level 2 amateur packet repeater, there is an additional address sub-field added to the end of the address field. This additional sub-field contains the call sign of the repeater to be used. This will allow more than one repeater to share the same rf channel, which has been a problem with the older protocols. If this field exists, the last octet of the source sub-field has its extender bit set to zero, indicating that more address-field data follows. The repeater address sub-field is encoded in the same manner as the destination and source address sub-fields, except for one bit in the last octet, called the "H" bit. The H bit is used to indicate whether a frame has been repeated or not. This is necessary to prevent someone from potentially receiving two identical frames, the one going to the repeater, and the one coming back from the repeater. Fig. 4 shows how the repeater address sub-field is encoded. Appendix B is an example of a complete frame on its way back from a repeater.

Octet	ASCII	Bin.Data	Hex Data
A15	W	10101110	AE
A16	B	10000100	84
A17	4	01101000	68
A18	J	10010100	94
A19	F	10001100	8C
A20	I	10010010	92
A21	SSID	HRRSSID1	

Bit Order --> 76543210

Fig 4. Repeater Address Encoding

Where:

1. The top octet is the first octet sent, with bit 0 being sent first, bit 7 sent last of each octet.

2. As with the source and destination address sub-fields discussed above, bit 0 of each octet is the HDLC address extender bit, which is set to zero on all but the last address octet (A21) where it is set to one.

3. The "R" bits are reserved just like in the source and destination sub-fields.

4. The "H" bit is the has-been-repeated bit. It is set to zero on a non-repeated frame, and set to one by the repeater when the frame has been repeated.

It should be noted that some of the advantages of this addressing scheme are mentioned in Appendix C.

Control Field Formats

The control field is responsible for identifying what type of frame is being sent, and is also used to convey commands and responses from one end of the link to the other to maintain proper control over the link.

The control fields used in AX.25 use the CCITT X.25 control fields for balanced operation, with an additional control field taken from ADCCP to allow connectionless and round-table operation.

There are three general types of AX.25 frames. They are the Information frame (I frame), the Supervisory frame (S frame), and the Unnumbered frame (U frame). Fig. 5 shows the basic format of the control field associated with these types of frames.

Control Field Type	Control Field Bits 7 6 5	4	3 2	1	0
I Frame	N(R)	P/F	N(S)		0
S Frame	N(R)	P/F	S S	0	1
U Frame	M M M	P/F	M M	1	1

Fig. 5. Control Field Formats

Where:

1. Bit 0 is the first bit sent, bit 7 is the last bit sent of the control field.

2. N(S) is the send sequence number (bit 2 is the LSB).

3. N(R) is the receive sequence number (bit 6 is the LSB).

4. The "S" bits are the supervisory function bits, and their encoding is discussed below.

5. The "M" bits are the unnumbered frame modifier bits and their encoding is discussed below.

6. The P/F bit is the Poll/Final bit. Its function is described in more detail shortly.

Control Field Definitions

Information Frame Control Field

All I frames have bit 0 of the control field set to zero. N(S) is the sender's send sequence number (the send sequence number of this frame). N(R) is the sender's receive sequence number (the sequence number of the next expected received frame. These numbers are described in the section regarding flow control.

Supervisory Frame Control Field

Supervisory frames are denoted by having bit 0 of the control field set to one, and bit 1 of the control field set to zero. S frames provide supervisory link control such as acknowledging or requesting retransmission of I frames, and link level window control. Since S frames don't have an information field, the sender's send variable and the receiver's receive variable are not incremented for S frames.

Unnumbered Frame Control Field

Unnumbered frames are distinguished by having both bits 0 and 1 set to one. U frames are responsible for maintaining control over the link beyond what is accomplished with S frames. They are also responsible for the establishment and tearing down of the link. U frames also allow for the transmission and reception of information outside of the normal flow control. Some U frames may contain information fields.

Control Field Parameters

Sequence Numbers and Variables

Every AX.25 I frame shall be assigned a sequential number from 0 to 7. This will allow up to seven outstanding I frames per level 2 connection at a time.

Send State Variable V(S)

The send state variable is an internal variable that is never sent. It contains the next sequential number to be assigned to the next transmitted I frame. This variable is updated upon the transmission of each I frame.

Send Sequence Number N(S)

The send sequence number is found in the control field of all I frames. It contains the sequence number of the I frame being sent. Just prior to the transmission of the I frame, N(S) is updated to equal the send state state variable.

Receive State Variable V(R)

The receive state variable is an internal variable that contains the sequence number of the next expected received I frame. This variable is updated upon the reception of an error-free I frame whose send sequence number equals the present received state variable value.

Received Sequence Number N(R)

The received sequence number is in both I and S frames. Prior to sending an I or S frame, this variable is updated to equal that of the received state variable, thus implictly acknowledging the proper reception of all I frames up to and including N(R)-1.

Poll/Final (P/F) Bit

The P/F bit may be used in all types of frames. It is used in a command (poll) mode to request an immediate reply to a frame. The reply to this poll is indicated by setting the response (final) bit in the appropriate frame. Only one outstanding poll condition per direction is allowed at a time.

Control Field Encoding

Information Frame Control Field

The information frame control field is encoded as shown in Fig. 6. These frames are sequentially numbered to maintain control of their passage over the link level connection.

Control Field Bits

7 6 5	4	3 2 1	0
N(R)	P/F	N(S)	0

Fig. 6. I Frame Control Field

Supervisory Frame Control Field

The supervisory frame control fields are encoded as shown in Fig. 7. In AX.25, S frames are used only as responses to other frames.

Control Field Bits	7 6 5	4	3 2	1 0
Receive Ready RR	N(R)	P/F	0 0	0 1
Receive Not Ready RNR	N(R)	P/F	0 1	0 1
Reject REJ	N(R)	P/F	1 0	0 1

Fig. 7. S frame control Fields

Receive Ready (RR) Response

Receive Ready is used to do the following:

1. To indicate that the sender of the RR is now able to recieve more I frames.

2. To acknowledge properly received I frames up to, and including N(R)-1.

3. To clear a previously set busy condition created by an RNR command having been sent.

It should be noted that the status of the other side of the link can be requested by setting the poll bit.

Receive Not Ready (RNR) Response

Receive not ready is used to indicate to the sender of I frames that receiver is temporarily busy and cannot accept any more I frames. Frames up to N(R)-1 are acknowledged. Any I frames numbered N(R) and higher that might have been caught in between and not acknowledged when the RNR command was sent are NOT acknowledged.

The RNR condition can be cleared by the sending of a UA, RR, REJ, or SABM frame. The P/F bit can be used within the RNR frame to interrogate the status of the other side of the link.

Reject (REJ) Response

The reject frame is used to request retransmission of I frames starting with N(R). Any frames that were sent with a sequence number of N(R)-1 or less are acknowledged. Additional I frames may be appended to the retransmission of the N(R) frame if there are any.

Only one reject frame condition is allowed in each direction at a time. The reject condition is cleared by the proper reception of I frames up to the I frame that caused the reject condition to be initiated.

As with the other supervisory responses, the P/F bit may be used in the REJ frame.

Unnumbered Type Frames

Unnumbered frame control fields are either commands or responses. This standard follows X.25 as much as possible. The only deviation from X.25 is in the addition of the Unnumbered Information (UI) frame from ADCCP. X.25 is designed to work with in full-duplex systems with only one main device (DCE) and potentially many users (DTEs).

Amateur Radio packet systems differ greatly on both of these respects. Not only is Amateur Radio packet networking done in a half-duplex rf environment, but many DCE/DTE links many be sharing the same channel. Many amateurs have rejected the use of X.25 as a result of these problems. X.25 can easily be enhanced so that it will perform properly over amateur radio.

Fig. 8 shows the layout of U frames implemented within this standard.

Control Field	Type	Control Field Bits 7 6 5	4	3 2	1 0
Set Asynchronous Balanced Mode-SABM	Cmd	0 0 1	P	1 1	1 1
Disconnect-DISC	Cmd	0 1 0	P	0 0	1 1
Disconnected Mode DM	Res	0 0 0	P/F	1 1	1 1
Unnumbered Acknowledge-UA	Res	0 1 1	F	0 0	1 1
Frame Reject-FRMR	Res	1 0 0	F	0 1	1 1
Unnumbered Information-UI	Either	0 0 0	P/F	0 0	1 1

Fig. 8. U Frame Control Fields

Set Asynchronous Balanced Mode (SABM) Command

The SABM command is used to place 2 stations in the asynchronous balanced mode. This is a balanced mode of operation known as LAP B where DCEs and DTEs are treated as equals.

Information fields aren't allowed in SABM commands. Any outstanding I frames left when the SABM command is issued will remain unacknowledged.

Disconnect (DISC) Command

The DISC command is used to terminate a link session between two stations. No information field is permitted in a DISC command frame. Any outstanding I frames will remain outstanding.

Disconnected Mode (DM) Response

The disconnected mode response is sent whenever the DTE or DCE receives a frame other than a SABM while in a disconnected mode. It is sent to request a set mode command, or to indicate it cannot accept a connection at the moment. The DM response cannot have an information field.

A DCE or DTE in the disconnected mode will respond to any command other than a SABM with a DM response with the P/F bit set to 1.

Unnumbered Acknowledge (UA) Response

The UA response frame is sent to acknowledge the reception and acceptance of a U frame command. A received command is not actually processed until the UA response frame is sent. An information field is not permitted in a UA frame.

Frame Reject (FRMR) Response

The FRMR response frame is sent to report that for some reason the receiver of a command or information frame cannot successfully process that frame and that the error condition is not correctable by sending the offending frame again. Typically this condition will appear when a frame without an FCS error has been received with one of the following conditions:

1. The reception of an invalid or not

implemented command or response frame.

2. The reception of an I frame whose information field exceeds the agreed upon length.

3. The reception of an improper N(R). This usually happens when the N(R) frame has already been sent and acknowldeged, or when N(R) is out of sequence with what was expected.

4. The reception of a frame with an information field where one is not allowed, or the reception of an U or S frame whose length is incorrect.

When a CMDR or FRMR frame is sent, an information field is added to the frame that helps to explain where the problem occurred. This information field is three octets long and its contents is shown if Fig. 9 below.

```
----------------------------------------------------------
|              Info mation Field Bits                     |
| 2 2 2 2 1 1 1 1 1 1 1 1 1 1                             |
| 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0         |
|---------------------------------------------------------|
| 0 0 0 0|Z|Y|X|W|  V(R) |C|  V(S) |0| Rejected Frame     |
|        | | | | |       | |       | | Control Field      |
----------------------------------------------------------
```

Fig. 9. FRMR Frame Information Field

Where:

1. The rejected frame control field carries the control field of the frame that caused the reject condition. It is in bits 1-8 of the information field.

2. V(S) is the current send state variable of the device reporting the rejection (bit 10 is the low bit).

3. V(R) is the current receive state variable of the device reporting rejection (bit 14 is the low bit).

4. If W is set to 1, the control field received was invalid or not implemented.

5. If X is set to 1, the frame that caused the reject condition was considered invalid because it was a U or S frame that had an information field that is not allowed. Bit W must be set to 1 in addition to the X bit.

6. If Y is set to 1, the information field of a received frame exceeded the maximum capacity of the device reporting the condition.

7. If Z is set to 1, the control field received and returned in bits 1 to 8 contained an invalid N(R).

8. Bits 8, and 20 to 23 are set to 0. Bit 12 is set to 0 if the rejected frame was a command, or 1 if if it was a response.

Unnumbered Information (UI) Frame

The unnumbered information frame is used to pass information along the link outside the normal information controls. This allows information fields to go back and forth on the link bypassing flow control. Since these frames are NOT acknowledgeable, if one gets wiped out, there is no way to recover it.

The UI frame is not defined in X.25. It has been taken from ADCCP to allow uncontrolled information to flow thru the link without interfering with a next higher layer.

Link Error Recovery

There are several link-level errors that are recoverable without tearing down the connection. These error situations may occur as a result of malfunctions within the DTE or DCE, or if transmission errors occur.

Invalid Frame or FCS Error

If an invalid frame is received, or a frame is received with an FCS error, that frame will be discarded with no action taken.

Device Busy Condition

When a DTE or DCE becomes temporarily busy, such as when receive buffers are full, it will send a receive not ready (RNR) frame. This tells the other side of the link that the device cannot handle any more I frames at the moment. This condition is usually cleared by the sending of a UA, RR, REJ, or SABM command frame.

Send Sequence Number Error

If the send sequence number , N(S), of an otherwise error-free received I frame does not match the receive state variable, V(R), a send sequence error has occured, and the information field will be discarded. The receiver will not acknowledge this frame, or any other I frames until N(S) matches V(R).

The control field of the erroneous I frame(s) will be accepted so that link supervisory functions can still be performed, such as checking the P/F bit. Because of this updating, the retransmitted I frame may have an updated P bit and N(R).

Reject (REJ) Error Recovery

REJ is used to request a retransmission of I frames following the detection of a sequence error. Only one outstanding reject condition is allowed at a time. This condition is cleared when the requested I frame has been received.

A device receiving the REJ command will clear the error by sending over the I frame indicated in N(R) of of the REJ command frame.

Time-out Error Recovery

When a transmission abnormality wipes out a single I frame, or the last I frame of a group, there is no way of telling this immediately, since the receiver does not necessarily know something was sent until another frame is sent resulting in an out-of-sequence error. To cope with this situation better, some form of time-out delay will be incorporated by the sender after it sends out a frame. This time-out timer is started at the time a frame is sent, and stopped by the reception of an acknowledgement for the sent frame. If the timer times out before an acknowledgement is received, any unacknowledged frames are retransmitted. The delay is an agreed-upon amount that will vary with the type of rf medium and signaling speed used.

Rejection Error

A rejection error condition occurs when an error-free received frame has one of the following problems:

1. An invalid command or response control field.

2. An invalid frame format.

3. An Invalid N(R).

4. An information field that exceeds the maximum the device can accept.

Once a rejection error occurs, no more I frames are accepted (with the exception of the P/F bit still usable) until the error is resolved. The error condition is reported to the other side of the link by sending a FRMR response frame.

Primary/Secondary versus Balanced Operation

There are two basic classes of link-level connections. The first, known as Link Access Procedure (or LAP) is often called an unbalanced service where the DCE is considered the primary (or master) devices and the DTEs are considered secondary (or slave) devices. The second class of service is known as LAPB, Link Access Procedure Balanced. In this service both devices are treated as equals as far as connection requests and other types of commands. There is still only one DCE and potentially many DTEs, but both ends can command the link equally.

Primary/Secondary (LAP) Operation

LAP is the older style of link control,

where most of the intelligence was assumed to be in a large mainframe (the DCE) and the end users were just using smart terminals (the DTEs). Since network software can have a lot of overhead, it made sense at the time to put most of the overhead in the big computer, and just enough smarts to make the link work in the terminals.

Balanced (LAPB) Operation

LAPB is a slightly modified version of LAP. It has been changed to allow the two sides of a link to operate in a more balanced manner. In the official version of X.25 there is still only one DCE to potentially many DTEs, but the two can operate more as equals than master and slave.

LAPB is what this document describes for use over Amateur Radio packet networks. Even when there is a network controller overseeing the network operation, the balanced link procedure will enhance operation.

Connection Operation

In amateur radio network operations, it would be very helpful if one level 2 protocol would work with the various rf systems in use. An example of this is the difference in operation between a simple two-station link, and multiple stations operating thru a network controller. Obviously, when a network controller exists, it should be considered the DCE, while the other stations connecting to it would be the DTEs. A simple two-station connection is another matter. To this type of connection the station requesting a connection should always be considered the DTE, while the device that is receiving the connection request should operate as the DCE. This simple rule should eliminate any ambiguity that might otherwise occur under these conditions.

NOTE There are a couple minor changes from the official X.25 standard in the protocol recommended here. These changes are done only as absolutely necessary to work over the shared rf media. Since X.25 was written to work so that one DCE talked with many DTEs over a closed network, it cannot properly cope with a channel where there may be many DCEs linked to many DTEs. Some amateurs have thrown X.25 out because of this problem. It seems to take just a couple minor changes in the initial link set-up procedure to make X.25 work properly over amateur radio. Where these changes are made, both the original X.25 procedure and the recommended amateur procedure will be noted.

LAPB Procedures

The following describes the procedures used to set-up, use, and disconnect a balanced link between a DTE and DCE. These procedures have been taken from X.25 and conform very closely to that standard, except where it was necessary to change due to the radio enviroment.

Address Field Operation

All transmitted frames shall have address fields conforming to above-mentioned rules. All frames should have both the destination device and the source device addresses in the address field, with the destination address coming first. This will allow many links to share the same rf channel. The destination address is always the address of the station(s) to receive the frame, while the source address contains the address of the dveice that sent the frame. The destination address can be a group name or club call however, if point to multi-point operation is allowed. This will be discussed further under link operations.

LAPB Connection Establishment

When a device (either a DCE or DTE) wishes to connect to another device, it will send a SABM command frame to that device and start a time-out timer (T1). If the other device is there and able to connect, it will answer with a UA response frame and at the same time reset both of it's internal state variables (V(S) and V(R)). The reception of the UA response frame at the other end will cause the device requesting the connection to abort the T1 timer and set its internal state variables to 0 also.

If the other device doesn't respond before T1 times out, the device requesting the connection will re-send the SABM frame, and start T1 running again. This trying to establish a connection will continue until the requesting device has tried unsuccessfully a number of times. That number (N1) is variable, depending on the frequency of operation, type of transmission (eg. terrestial vs. satellite), and the signaling speed in use. N1 will be discussed in another section.

Information Transfer

Once a connection has been established as outlined above, both devices are able to accept I, S, and U frames.

Sending of I Frames

Whenever a station has an I frame to transmit, it will send the I frame with N(S) of the control field equal to it's current send state variable V(S). Once the I frame is sent, the send state variable is incremented by one.

The station should not transmit any more I frames if it's send state variable equals the last received N(R) from the other side of the link plus seven. If it were to send more I frames, the flow control window would be exceeded and errors could result.

If a device is in a busy condition, it may still send I frames as long as the other device is not also busy.

If a device is in the frame-rejection mode, it will stop sending I frames.

Receiving I Frames

If a device receives a valid I frame (one with a correct FCS and whose send sequence number equals the receiver's receive state variable) and is not in the busy condition, it will accept the received I frame, increment it's receive state variable, and act in one of the following manner:

1. If it has an I frame to send, that I frame may be sent with the transmitted N(R) equal to it's receive state variable V(R) (thus acknowledging the received frame. Alternately, the device may send an RR frame with N(R) equal to V(R), and then send the I frame.

2. If tnere are no outstanding I frames, the receiving device will send an RR frame with N(R) equal to V(R).

If the device is in a busy condition, it may ignore any received I frames without reporting this condition other than repeating the indication of the busy condition.

If a busy condition exists, the station receiving the busy condition indication should poll the sender of the busy indication periodically until the busy condition disappears.

The reception of I frames that contain zero length information fields shall be reported to the next level but no information field will be transferred.

When an I frame is received with a correct FCS, but it's send sequence number does not match the current receiver's receive state variable, the frame should be discarded and a REJ frame should be sent with a receive sequence number equal to one higher (modulo 8) than the last correctly received I frame . Any out-of-sequence received I frames should be handled in this manner. The received state variable and poll bit in such a discarded frame should be checked before throwing it away, and take any action needed depending on the condition of them.

Receiving Acknowledgement

Whenever an I or S frame is correctly received, even in a busy condition, the N(R) of the received frame should be checked to see if it includes an acknowledgment of outstanding sent I frames. The T1 timer should be reset if the received frame actually acknowledges previously unacknowledged frames. If the T1 timer is reset, and there are still some frames that have been sent that are not acknowledged, T1 should be started again. If the T1 timer runs out before an acknowledgement is received, the device should proceed to the retransmission procedure.

Receiving Reject

Upon receiving a REJ frame, the transmitting station will set its send state variable to the same value are the REJ frames received sequence number in the control field. The device will then retransmit any I frame(s) outstanding at the next available opportunity conforming to the following:

1. If the device is not transmitting at the time, and the channel is open, the device may commence to retransmit the I frame(s) immediately.

2. If the device is operating on a full duplex channel transmittiong a U or S frame when it receives a REJ frame, it may finish sending the U or S frame and then retransmit the I frame(s).

3. If the device is operating in a full duplex channel transmitting another I frame when it receives a REJ frame, it may abort the I frame it was sending and start retransmission of the requested I frames immediately.

4. The device may send just the one I frame outstanding, or it may send more than one if any more I frames followed the first one not acknowledged, provided the total to be sent does not exceed the flow control window (7 frames).

If the device recives a REJ frame with the poll bit set, it should respond with either an RR or RNR frame with the final bit set before retransmitting the outstanding I frame(s).

Receiving an RNR Frame

Whenever a device receives an RNR frame, it may transmit or retransmit the I frame whose send sequence number equals that of the received sequence number indicated in the RNR control field. If timer T1 runs out after the RNR was received, the waiting acknowledgement procedure listed below should be performed. The poll bit may be used in conjunction with S frames to test for a change in the condition of the busied out station. No I frames other than the one mentioned above may be sent out before the busy condition is cleared.

Sending a Busy Indication

Whenever a device enters a busy condition, it will indicate this by sending an RNR response at the next opportunity. While the device is in the busy condition, it may receive and process S frames, and if a received S frame has the P bit set to one, the device should send a RNR frame with the F bit set to one at the next possible opportunity. To clear the busy condition, the device should send either a RR or REJ frame with the received sequence number equal to the current receive state variable, depending on whether the last received I frame was properly received or not.

Waiting Acknowledgement

The device should maintain an internal retransmission count variable which is set to zero whenever another I frame is acknowledged (either thru the reception of a UA or RNR frame, or when a received I or S frame has an N(R) higher than the last received N(R), showing the acknowledgement of additional I frames).

Any time the timer T1 runs out, the device will re-enter the timer recovery condition, the retransmission count variable will be incremented by one, and another internal variable (X) will be set to the current send state variable value.

The device will then restart the T1 timer, set its receive state variable to the last receive sequence number, and retransmit the corresponding I or S frame with the P bit set to one.

The timer recovery condition is cleared when the device receives a valid S frame with the F bit set to one.

If the device receives an S frame with the F bit set to one and N(R) within the range from the current send state variable to X mentioned above inclusive while in the timer recovery condition, this condition will be cleared, and the send state variable will be set to the N(R) received.

If the device receives an S frame with the F bit set to zero but otherwise the same condition as the last paragraph, the timer recovery condition will NOT be cleared. The received N(R) may be used however to update the send state variable. The device may keep the last I frame transmitted (even if it was acknowledged) to be retransmitted with the P bit set to one if timer T1 expires at a later time.

Once the retransmission count variable reaches N2, the device should proceed to the resetting procedures outlined below.

Link Disconnection

When in the information-transfer phase, either device may initiate a link disconnection by sending a DISC frame. It should then start its T1 timer, and wait for a response. If the proper response doesn't come before T1 times out, it should send the DISC frame again and restart T1. If this happens N2 times, the device should enter the disconnected state.

When a DISC frame is received, the receiver should return a UA response frame, and enter the disconnected state.

Disconnected State

After having sent a DISC frame and received a UA, or receiving a DISC and having sent a UA, the device will enter the disconnected state.

In the disconnected state, the device may initiate a link set-up as outlined in connection establishment above. It may also respond to the reception of a SABM and establish a connection, or it may ignore the SABM and send a DM instead.

Any station receiving a DISC command while in the disconnected state should send back a DM response frame.

Any device receiving a command frame other than a SABM or UI frame with the P bit set to one should respond with a DM frame with the F bit set to one. The offending frame should also be ignored.

When the device enters the disconnected state after an error condition or if it has recovered from an internal error condition by coming up in the disconnected state, it should indicate this by sending a DM response rather than a DISC frame. It should start the T1 timer when the DM is sent, and if T1 times out before getting a SABM or DISC frame back, it should send another DM frame, and restart T1. After retransmitting the DM frame N2 times, the device will remain in the disconnected state, and no other action will be taken.

Resetting Procedure

The resetting procedure is used to initialize both directions of flow after a nonrecoverable error has occured. This resetting procedure is only used when in the information transfer phase of an AX.25 link.

A device shall request a reset by sending an SABM frame. Upon receiving an SABM frame from a station previously connected to, the receiver of an SABM frame should send a UA frame back at the earliest opportunity. Both devices should then set their send and receive state variables to zero. Any busy condition that previously existed will also be cleared.

It is possible to initiate a disconnect procedure instead of resetting the link.

One device may ask the other to reset the link by sending a DM response frame. After the DM frame is sent, the sending device will then enter the disconnected state.

One device may ask the other to initiate a link reset by transmitting a FRMR response frame.

After sending the FRMR frame, The sending device will enter the frame reject state. This condition is cleared when the device that sent the FRMR frame receives an SABM or DISC command, or a DM response frame. Any other command received while the device is in the frame reject state will cause another FRMR to be sent out with the same information field as originally sent.

The device that sent the FRMR frame should start the T1 timer when the FRMR is sent. If above mentioned frames are not received before the timer runs out, the FRMR frame should be retransmitted, and the T1 timer restarted as described in the waiting acknowledgement section above. If the FRMR is sent N2 times without success, the link should be reset.

Rejection Conditions

A device should initiate the link-reset procedure when a frame is received with the correct FCS and address field during the information transfer phase with one or more of the following conditions:

1. The frame is not known as a command or response to the device.

2. The information field is invalid (as an example is longer than 256 octets).

A device will initiate a reset procedure whenever it receives a DM or FRMR response frame during the information transfer phase.

A device may initiate a reset procedure also whenever it receives a UA response frame or if it receives an unsolicited response frame with the F bit set to one.

Collision Recovery

Collisions in a Half-Duplex Enviroment

Collisions of frames of any type in a half-duplex enviroment are essentially taken care of by the retry nature of the T1 timer and retransmission count variable. No other special action needs to be taken.

Collisions in a Full-Duplex Enviroment

Collisions in a full-duplex enviroment are not really frame collisions, but have more to do with the devices being pulled in two different directions at the same time.

Collisions of Unnumbered Commands

If the sent and received U command frames are the same, both devices should send a UA response at the earliest opportunity, and both devices should enter the indicates state.

If the sent and received U commands are different, both devices should enter the disconnected state, and transmit a DM frame at the earliest opportunity.

Collision of a DM with a SABM or DISC

When an unsolicited DM response frame is sent, a collision between it and a SABM or DISC may occur. In order to prevent this DM from being misinterpreted, all unsolicited DM frames should be transmitted with the F bit set to zero. All SABM and DISC frames should be sent with the P bit set to one, so there isn't any confusion when a DM frame is received.

Connectionless Operation

In Amateur Radio circles, there is a type of operation that isn't really feasable using level 2 connections. This operation is the roundtable, where several amateurs may be engaged in one conversation. The only way to accomplish this in a connected mode would be to have everyone cross-connected with each other. This would require a seperate frame to be sent to each member of the roundtable every time someone says something. Obviously, this mode is not practical. The way most amateur packet radio enthusiasts have ended up implementing the roundtable operation is outside the AX.25 connection, but still using the AX.25 frame structure. AX.25 does allow a special frame for this operation, called the Unnumbered Information (UI) frame. It is recommended that when this type of operation is in use, the destination address have a code word installed in it to prevent the users of that particular roundtable from seeing all frames going thru the shared RF medium. An example of this is if a group of amateurs are in a roundtable discussion about packet radio, they could put PACKET in the destination address, so they would only receive frames from others in the same discussion. An added advantage of the use of AX.25 in this manner is that the source of each frame is in the source address sub-field, so software could be written to automatically display who is making what comments.

Admittedly, this is a kludge to the level 2 AX.25 protocol. THis type of operation really belongs at the next layer (layer 3, packet level) of operation, but until layer 3 is implemented, this appears to be an acceptable substitute.

Keep in mind that this mode is connectionless, so all transmitted frames should be of good quality, as there will be no requests for retransmissions of bad frames. Collisions will also occur, with the potential of losing the frames that collided.

List of System Defined Parameters

Timers

It is recommended that there are two timers used to maintain the integrity of the AX.25 layer 2 connection.

The first timer, T1, is used to make sure a device doesn't wait forever for a response to a frame it sends. This timer cannot be expressed in absolute time, since the time required to send frames varies greatly with the baud rate used at level 1. T1 should be at least twice the time it would take to send a maximum length frame to the other end of the link, and get the proper response frame back from the other end of the link. This would allow time for the other end of the link to do some processing before responding.

The second timer, T2, is used whenever T1 isn't running to make sure that a supervisory frame is sent periodically to maintain link integrity. It also will vary dramatically depending on layer 1 constraints, and is subject for further study.

Maximum Number of Retrys (N2)

The maximum number of retrys is used in conjunction with the T1 timer. It will vary depending on the layer 1 in use, but will generally be sixteen.

Maximum Number of Octets in an I Field (N1)

The maximum number of octets allowed in the I field will be 256. There should also be an even multiple number of octets.

Maximum Number of I Frames Outstanding (k)

The maximum number of outstanding I frames at a time is seven. A smaller number may be used at any time, provided it is agreed upon ahead of time.

First
Bit Sent

Flag	Address	Control	FCS	Flag
Ø111111Ø	112/168 Bits	8 Bits	16 Bits	Ø111111Ø

Fig. 1A. U and S Frame Construction

First
Bit Sent

Flag	Address	Control	PID	Info.	FCS	Flag
Ø111111Ø	112/168 Bits	8 Bits	8 Bits	N*8 Bits	16 Bits	Ø111111Ø

Fig. 1B. Information Frame Construction

First
Octet Sent

Address Field of Frame	
Destination Address	Source Address
A1 A2 A3 A4 A5 A6 A7	A8 A9 A1Ø A11 A12 A13 A14

Fig. 2A. Non-Repeater Address Field Encoding

First
Octet Sent

Address Field of Frame		
Destination Address	Source Address	Repeater Address
A1 A2 A3 A4 A5 A6 A7	A8 A9 A1Ø A11 A12 A13 A14	A15 A16 A17 A18 A19 A2Ø A21

Fig. 2B. Repeater Address Field Encoding

Appendix A. Non-Repeater Frame Example

Octet	ASCII	Bin.Data	Hex Data
Flag		01111110	7E
A1	K	10010110	96
A2	8	01110000	70
A3	M	10011010	9A
A4	M	10011010	9A
A5	0	10011110	9E
A6	space	01000000	40
A7	SSID	01100000	60
A8	W	10101110	AE
A9	B	10000100	84
A10	4	01101000	68
A11	J	10010100	94
A12	F	10001100	8C
A13	I	10010010	92
A14	SSID	01100001	61
Control	SABM	00111111	3F
PID	none	11110000	F0
FCS	part 1	XXXXXXXX	HH
FCS	part 2	XXXXXXXX	HH
Flag		01111110	7E

The frame shown is an SABM frame, not going thru a level 2 repeater, from WB4JFI (SSID=0) to K8MMO (SSID=0), with no level 3 protocol.

Appendix B. Repeater Type Operation

Octet	ASCII	Bin.Data	Hex Data
Flag		01111110	7E
A1	K	10010110	96
A2	8	01110000	70
A3	M	10011010	9A
A4	M	10011010	9A
A5	0	10011110	9E
A6	space	01000000	40
A7	SSID	01100000	60
A8	W	10101110	AE
A9	B	10000100	84
A10	4	01101000	68
A11	J	10010100	94
A12	F	10001100	8C
A13	I	10010010	92
A14	SSID	01100000	60
A15	W	10101110	AE
A16	B	10000100	84
A17	4	01101000	68
A18	J	10010100	94
A19	F	10001100	8C
A20	I	10010010	92
A21	SSID	11100011	E3
Control	SABM	00111111	3F
PID	none	11110000	F0
FCS	part 1	XXXXXXXX	HH
FCS	part 2	XXXXXXXX	HH
Flag		01111110	7E

The above frame is the same as in Appendix A, but with the addition of a repeater address sub-field (WB4JFI, SSID=1). The H bit is set, indicating this is from the output of the repeater.

Appendix C.

Advantages of the WB4JFI Addressing Scheme

Some of the advantages to using this addressing system are:

1. Every packet station will have a unique fixed address that doesn't change every time a new network is logged into.
2. Relocating to a new area won't cause major (or minor) problems.
3. Allows for more than 62 or 31 users at a time.
4. No local packet guru is needed to assign addresses with attendant concerns of backup and transfer during failure.
5. Direct or network operation requires no change of address.
6. All the problems with dynamic allocation/de-allocation are eliminated.
7. Reduces local co-network interference due to users in overlapping local network rf domains with the same address fields.
8. With every frame having both the destination and source addresses in them, it will be a lot easier to set-up and run multiple connections on the same data channel without having problems arise as to who is sending what frames to whom.
9. In round-table operation, every frame sent will have the source address imbedded in it, allowing automatic printing of the source of the frame.

Appendix D. Layer 2 AX.25 State Table

State	I with Poll	I without P	RR with Final	RR without F	REJ with Final	REJ without F	RNR with Final	RNR without F	SABM either	DISC either	UA either	DM either	FRMR either
S1 Disconnected	DM		DM		DM		DM		UA,S5	DM		SABM,S2	
S2 Link Setup									UA	DM,S1	S5	S1	
S3 Frame Reject	FRMR		FRMR		FRMR		FRMR		UA,S5	UA,S1		SABM,S2	SABM,S2
S4 Disconnect Rqst									DM	UA	S1	S1	
S5 Information Xfr	RR	I	I	I	I	I	S9	S9	UA	UA,S1	SABM,S2	SABM,S2	SABM,S2
S6 REJ Frame Sent	RR,S5	I,S5	I	I	I	I	S15	S15	UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2
S7 Waiting Acknow.	RR	I	I,S5	I	I,S5	I	S9	S12	UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2
S8 Device Busy	RNR	RNR	I	I	I	I	S10	S10	UA	UA,S1	SABM,S2	SABM,S2	SABM,S2
S9 Remote Device Busy	RR	RR	I,S5	I,S5	I,S5	I,S5			UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2
S10 Both Devices Busy	RNR	RNR	I,S8	I,S8	I,S8	I,S8			UA,S8	UA,S1	SABM,S2	SABM,S2	SABM,S2
S11 Waiting Acknow. and Device Busy	RNR	RNR	I,S8	I	I,S8	I	S10	S13	UA,S8	UA,S1	SABM,S2	SABM,S2	SABM,S2
S12 Waiting Acknow. and Remote Busy	RR	RR	I,S5	I,S7	I,S5	I,S7			UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2
S13 Waiting Acknow. Both Devices Busy	RNR	RNR	I,S8	I,S11	I,S8	I,S11			UA,S8	UA,S1	SABM,S2	SABM,S2	SABM,S2
S14 REJ Sent and and Device Busy	RNR	RNR	I	I	I	I	S16	S16	UA,S8	UA,S1	SABM,S2	SABM,S2	SABM,S2
S15 REJ Sent and Remote Busy	RR,S9	RR,S9	I,S6	I,S6	I,S6	I,S6			UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2
S16 REJ Sent and Both Devices Busy	RNR	RNR	I,S14	I,S14	I,S14	I,S14			UA,S5	UA,S1	SABM,S2	SABM,S2	SABM,S2

LEVEL 3 POSITION PAPER

Terry Fox, WB4JFI
Vice-President, AMRAD
1819 Anderson Road
Falls Church, VA 22043

Introduction

Now that the amateur packet radio community seems to have agreed on a protocol at layer 2, the link level of the Open System Interface Reference Model (OSI-RM), it appears that it is time to begin work on the layer 3, or network layer. Layer 3 is actually made up of two sub-layers, a local or metropolitan network sub-layer, and an internetwork sublayer.

The local area network is responsible for the proper transfer of packets among a group of local users. The term local can be misleading, as a local network could actually be a network operating on hf, where the participants are actually spread out over a large geographical area. A local network is generally considered a group of devices interconnected directly together at the layer immediately above the link layer. These devices may be corresponding directly, or they may be operating thru an intermediary, such as a local (or metropolitan) network controller.

The internetwork sub-layer is half a step above the local network, and is used to interconnect individual local networks. This allows a user on one local network to communicate with another user on a different local network. Depending on how the internet sublayer operates, another layer above it (layer 4, transport layer) may be required to assist in the re-assembly of data sent over the internet layer.

Types of Network Operation

There are two basic types of networks, both at the local and internet sublayers. One is the connection type, and the other is the datagram type.

Connection Type Networks

The connection type of network requires a connection be established and maintained between the two devices wishing to communicate before any data can be transferred. The connection looks very much like the HDLC type layer 2 links that are established before I frames may be passed along the link.

The connection network is like a small town telephone company. Whenever a local call is made, as long as it is between the same two people, the connection will be made the same way every time (usually for a different reason though, because that's the only way a connection can be made between the two people, not necessarily the best way). In a connection network, once the connection is made, all frames MUST follow the same path. If anything should happen to that path, the connection must be torn down and re-established over again.

The main advantages of the connection type of network are:

1. Once the connection is established, very little overhead is required to maintain proper operation.

2. Generally, a connection oriented network does not allow data packets to be received out of their proper sequence, thereby greatly simplifying the transport layer required.

Some of the disadvantages of the connection type of network are:

1. Once a connection is established, all packets must follow the path generated while the connection was being made. This could be a problem if either the network or one of the devices involved are marginal in nature.

2. Out of sequence packets are not usually allowed, meaning a valid packet may have to be retransmitted because an earlier packet got lost.

Datagram Type Networks

A datagram type network operates in a different manner than a connection type network. Each packet in a datagram network contains a header that should have all the information necessary to get it from its source to its destination totally independently of all packets sent before or after it. A datagram network is like sending a bunch of letters to the same person on different days from the same post office. Just because the same post office was used, and the addressee is the same, doesn't mean all letters follow the same path from you to the destination.

The basic advantages and disadvantages of a datagram network are just the opposite as those of the connection network. While every packet can be routed a different way (potentially going around trouble spots in the network), the added size of each packet (due to a larger header) and the added complexity of the transport layer (to re-align out of sequence packets) add up to more overall complexity in the software or hardware used to implement a datagram network.

History

When AMRAD first started looking into the layers higher than layer 2, we were sold on the datagram type of network. It seemed to us that the amateur radio enviroment that a network must operate in can become very unreliable (not only because the rf medium may vary dramatically, but also because of the voluntary nature of the amateurs participating). Datagrams can find their way from on end of a network to the other no matter how convoluted the network may become due to equipment or operator failure, as long as there is at least one good path to the destination.

Our initial decision to use a datagram network was quickly tempered however, when we found out how large a program would be required to handle datagrams properly. We couldn't find anyone who was operation a network level datagram service without having implemented in a higher-level language on a large maniframe computer (or at the very least a mini). Obviously, we weren't going to write a program of this size in the not too distant future.

When AMRAD got together with the New Jersey packet contigent to discuss the level 2 AX.25 protocol, we met at Telenet with Eric Scace, K3NA. Eric has worked with X.25 for quite a while (he was involved in the writing of the X.25 recommendation), and he was able to bring to our meetings invaluable insight into the inner workings of X.25. I found out there is a hugh difference between reading a protocol specification, and talking to someone about that protocols actual implementation. Eric was able to convince us that a connection oriented network such as X.25 could be implemented properly in an amateur radio environment. He also halped us decide how to implement the X.25 level 3 protocol properly, and how to add information necessary to make the protocol operate in an Amateur Radio enviroment while still maintaining the integrety of the X.25 protocol specification. Fortunately, Eric was able meet with the packet crowd at the AMSAT sponsored get-together last October. I think that a lot of questions were answered at that meeting, and although the rest of the crowd isn't convinced that X.25 is the proper way to go at layer 3, we at AMRAD have decided to go with X.25 for our local network. Gordon Beattie Jr., N2DSY, from The Radio Amateur Telecommunications Society (the New Jersey group mentioned above) is in the process of writing up the layer 3 AX.25 protocol specification now, and it will be available soon.

Basically, AX.25 level 3 follow the CCITT X.25 level 3 protocol having to do with virtual circuits. All references to datagram and permanent virtual circuits are to be ignored.

We have added two amateur radio network facilities. These facilities allow for either **explicit** route selection or **implicit** route selection. Explicit route selection is where the requesting station describes exactly the route the connection should take. Implicit route selection is where the requesting station describes where the station is, and the actual path is determined by the network.

Unfortunatly, the level 3 protocol is just too complex to present in a paper of this type, so if you are interested in AX.25 layer 3 details, I suggest you get in touch with AMRAD or Gordon Beattie, and we will keep you advised as the protocol develops.

A NEW PACKET-RADIO CONTROLLER BOARD

Terry Fox, WB4JFI
Vice-President, AMRAD
1819 Anderson Road
Falls Church, VA 22043

Abstract

This paper describes the AMRAD packet assembler/disassembler (PAD) to be released soon. It is Zilog Z80A based, uses a Zilog 8530 serial communications controller and is packaged on an S-100 pc board.

Introduction

One of the main problems with packet radio is that until recently there hasn't been a lot of hardware to support the various protocols being used. Except for a few pockets of activity, most of North America has adopted the idea of using a separate board (usually a single-board computer) to handle the actual generation and reception of frames. The first production board available to the amateur to do this was the Vancouver Amateur Digital Communication Group's Terminal Node Controller (TNC). This board was (and still is) available primarily as a bare board which had to be built up by the packet enthusiast. The board uses an Intel 8085 processor, 4k each of EPROM and static RAM, a serial or parallel device to communicate to your terminal/computer, and an Intel 8273 HDLC controller. The VADCG TNC moved the packet-radio software from the host computer to a separate board, and at the same time allowed many people to use a simple terminal with packet radio.

The next board that came out was designed by and is available from the Tuscon Amateur Packet Radio Corporation (TAPR). Its basic design philosophy is the same as the VADCG TNC in that it also handles all of the frame-level generation and reception of packets, requiring only a terminal or serial/parallel interface to a computer. Its actual hardware design is quite a bit different from the VADCG TNC, however. In addition to a different CPU (a Motorola 6809), it boasts quite a bit more memory (six byte-wide RAM/EPROM sockets normally fitted with 24k of EPROM and 6k static RAM), a different HDLC controller chip (Western Digital 1933), timed interrupts, a non-volatile memory, and a complete Bell 202 compatible modem (using the Exar modem chips). The TAPR group had time to study the VADCG TNC and made a lot of improvements when they designed their board. Another advantage to the TAPR TNC is that it is purchased as an assembled board, reducing greatly the chances of failure for the user. One of the many problems with the TAPR TNC actually has nothing to do with the board itself. TAPR has had a lot of problems getting the boards designed and into production. As of when this paper is being written, TAPR is getting the boards out to the last of their customers. This is one of the disadvantages of being out at the infamous leading edge. This also shows that the TAPR group doesn't want to send out TNCs that aren't as good as they can be, and the boards are definitly worth the wait.

The Amateur Radio Research and Development (AMRAD) group has been watching the progress of these boards with interest for quite a while now, and we figured it was time we got into the act. Most of us in AMRAD that are into the development stages of packet radio use S-100 based systems, usually with Z80 processors. So the TAPR TNC is rather difficult for us to write software for. Also, while having the modem on the TNC is nice for two-meter operation. However, when testing new ideas out for hf operation or otherwise when a different modem is needed, all the on-board modem does is take up board space.

We had some different problems with the VADCG TNC. The primary one is that the memory (both RAM and EPROM) are too small. I have modified the EPROM space to allow the use of either 2716's (up to 8k EPROM) or 2732's (up to 16k EPROM), but not many people want to take an X-ACTO (tm) knife to their board. Bill Danielson, N6FQR has modified the VADCG TNC to run 2k static RAM chips and also be able to download software from another computer to the TNC during software debugging, but here again, the board must be butchered.

We have come up with an S-100 board that contains an Intel 8273 protocol chip and some support logic so that we can write and debug software for the VADCG TNC in our S-100 systems before blowing any EPROMs for the VADCG TNC. This system works very nicely when coming up with changes in existing software, or working on some radical new idea without having to reburn EPROMs for every failure (no, I haven't written error-free code in a long time).

An additional problem with the VADCG TNC is that the baud rate on the packet channel is hardware selected and it goes down only to 600 bauds. Another modification was made to allow the baud rate to be software selectable and to allow the slower speeds needed for hf operation.

The AMRAD PAD

The further along we went with packet radio, the more kludges we needed to make on the VADCG TNC to allow us to do the experimentation we wanted. Since the TAPR TNC was in the initial design stages and using a processor we weren't accostomed to, it became apparent to us that it would be easier to design and build a whole new TNC board.

After making the decision to design our own board, we next had to decide what to put on the board, and what physical size it should be. I'm sure that it will come as a surprise to almost no one that the board will fit into an S-100 frame, and steal its power off the S-100 bus. This does not preclude the possibility of using the board stand alone with a single S-100 edge connector to supply power if the user isn't using an S-100 system.

Basic PAD Layout

Fig. 1 shows the basic layout of the AMRAD PAD (PAD stands for Packet Assembler/Disassembler). The PAD was designed to be very flexible. In addition to allowing the user to connect to it in either serial or parallel mode (the other boards do this also), it allows for fully adjustable baud rate on both serial ports, and if necessary, both serial ports can be programmed to be HDLC channels. It also has room for controlling the speed of a multi-speed programmable modem. The large amount of memory allows for downloading and debugging of programs in the PAD rather than needing another simulator board in a larger microcomputer. The large RAM space also allows a lot more space for buffers and other storage, meaning that the PAD should be able to run more than one connection per HDLC channel (something that could come in handy in the near future). The EPROM area can programmed to accept 2716, 2732, or 2764 devices, allowing plenty of room for expansion. A detailed description of the PAD board follows.

PAD Power Supply Circuitry

The power necessary to operate the PAD board is supplied thru the S-100 bus connector on the bottom of the board. The PAD uses three voltages, +8V at about an amp, +18V at about 50 milliamps, and -18V at approximately 100 milliamps. These voltages are regulated on the board to supply the +5 volts and ± 12 volts that the rest of the board requires. The five-volt bus has two 7805, TO-220 type voltage regulators, one on each side of the board. The ± twelve-volt regulators use the smaller, TO-92 style packages. These voltages are used primarily for the RS-232 driver chips and the real-time clock.

In addition to the power supply mentioned above, there is also a battery supply on board to run the real-time clock, the standby RAM chip, and some of these devices support logic. The PAD also has a sense circuit on board to tell the clock chip when to go into the standby mode because the power is being shut down. This is accomplished by using a micro-power op amp that monitors both the battery voltage and the +5-volt bus, and sends an active low signal out when the main power is below the battery.

Z80 CPU and Support Circuitry

The PAD board is designed around the Zilog Z80 processor. The master oscillator is an Intel 8080 support IC, the 8224, running at 18.432 MHz. This frequency was chosen because it is a multiple of the common baud-rate frequency of 1.8432 MHz, and also because it is high enough to be used in the refresh logic for the dynamic memory. The master oscillator is divided by five to produce a 3.6864-MHz clock for the Z80 CPU, along with the rest of the board. This frequency is a multiple of the 1.8432-MHz clock desired, so it can be used as the clock input to the serial-interface chip.

The Z80 reset logic consists of one half of a dual D flip-flop to make sure that all devices using reset get a properly timed signal. The Z80's NMI pin is optionally connected to a pushbutton to allow a way of interrupting CPU operations for debugging.

The Z80's RD and WR signals are buffered, since they are fed to many of the other devices on board. IORQ and M1* are ORed together to produce INTA* that is used by the 8530 interrupt support logic.

EPROM Memory and Support Logic

There are four 28-pin sockets provided for EPROMs on the PAD board. They have been placed so that Textool Zero-Insertion-Force (ZIF) sockets can be used if the EPROMs are going to be changed frequently (they are recommended). A jumper area is provided to allow the EPROM space to be programmed for 2716, 2732, or 2764 type EPROMs. If 2716's are used, 6116 type RAM chips also can be plugged into the EPROM sockets, allowing up to 62k of RAM and one loader EPROM to be used. This mode of operation is advantageous when testing software written on a larger system and downloaded to the PAD. It also comes in handy when debugging the dynamic RAM, since the static RAM in the EPROM socket will allow monitors or memory testers to run properly.

A 74LS138 decoder is used to provide the chip enable to the EPROMs. There is also a signal generated called PROM*, which is used to automatically deselect the EPROM memory space from the dynamic memory.

Dynamic Memory and Support

The dynamic memory consists of eight 4164 type chips for a potential of 64k of memory. When this board was in its initial design stage, 4116 type devices were used since they provided a lot of memory for a very little amount of money and board space. As the PAD design progressed, the prices of the 64k chips dropped down to where it became cost effective to put them on instead. The refresh support logic is the same for both types of devices, and the power supply considerations are simpler for the 64k chips. The 64k devices also generate less noise on the power bus.

The sixteen address lines from the CPU are multiplexed onto the eight address lines of the 4164's with 74LS157 multiplexers driven by a select timing signal. This select signal, RAS*, and CAS* signals are generated by three D-type flip flops that are driven by the two clocks mentioned earlier, M1*, MREQ*, and RFSH* signals from the Z80 CPU, and some additional logic. The PROM* signal mentioned earlier is added into the refresh logic to deselect the dynamic memory when an EPROM might be addressed instead of RAM.

The refresh logic is of standard design taken right out of the Zilog dynamic memory interface application note. The only differences are the addition of the two added address lines into the address multiplexers, and the use of a times five clock instead of a times two clock. This higher clock rate is actually better because it causes the refresh signals to be within tolerences where a times two signal would cause some timing signals to be slightly out of specification.

Input/Output Decode Logic

Most of the port decode logic consists of one IC, a 74LS138 decoder. It sends chip-select signals to the various I/O devices, except for the real time clock and standby RAM chips. Because the real-time clock and stand-by RAM chips use many more ports than the other devices, they have separate chip-enable logic. The following is a brief chart of the port assignments on the PAD:

HEX ADDRESS	DEVICE
00-03	8530 SCC
04-07	Terminal PIO
08-0B	Auxilliary PIO
0C-0F	Clock Interrupt Latch
10-13	Standby RAM Latch
40-5F	Real Time Clock Chip
80-FF	Standby RAM Chip

8530 and Support Logic

The two serial channels are generated by a relatively new device to the amateur packet enthusiast, the Zilog 8530 Serial Communications Controller, or SCC. When the PAD was in its initial design, it used the Intel 8273 protocol chip. This was the chip that most of us were familiar with, and software already existed to drive the 8273 properly. When we discovered the 8530, we realized that it could greatly simplify the PAD. Not only does it have two completely programmable channels, it also has two separatly programmable baud-rate generators, and two DPLLs. This one chip can support both the HDLC packet channel and the serial channel to the terminal or computer. In addition, if the terminal or computer is interfaced to the PAD board thru the parallel port, both serial channels can be used as HDLC packet channels. This might be useful when the PAD board is used with a larger computer as a gateway between two networks, or if one channel is assigned permanently to one type of operation (say hf), and the other is used for a radically operation (say vhf or phone line interface to Telenet or ARPA).

Even though the SCC is designed by the same company as the Z80 and it is supposed to be compatible, there are some timing problems between the SCC and the CPU. The main problem has to do with interrupt-acknowldegement timing. The SCC is designed to work on the Zilog Z-BUS, and its daisy-chain interrupt structure is slightly different than the Z80 peripherals used on the rest of the PAD board. This timing problem is cured by adding a couple chips that are used to extend the time of an interrupt-acknowldegement cycle, and also allow the SCC to be properly reset by hardware. The timing of this delay is accomplished by waiting the Z80 CPU with a delayed signal from a 74LS164 shift register. As with the dynamic refresh logic mentioned above, this modification is right out of an application note from Zilog.

Since both of the serial channels can be used as HDLC channels, it was decided to put timers on the RTS signal of both channels. These are 555 type timer designed to time out in about 30 seconds and stay off until reset by RTS changing back to an inactive state. The timer output of each of these time-out circuits are fed back into the CPU thru two bits on the auxilliary PIO, allowing the PIO to be programmed to generate an interrupt anytime the timer changes state. Not only does this give a fairly positive indication that the transmit command was successfully accomplished, but it also gives the CPU an indtcation if a time out has occured, and allows for a fairly graceful recovery (as opposed to hitting the reset button) after a time-out situation. These timers can be jumpered around in both channels if they aren't necessary.

The auxilliary PIO output side generates two signals that can be jumpered into the transmit data signal from the SCC. These signals are useful for adding a cw i-d by changing the connected modem between mark and space independantly of the SCC. This means a cw i-d can be generated without having to change the SCC's operating mode and kludging the software as is done on the VADCG TNC. Hopefully this will simplify both the SCC support software and the cw i-d routine.

The rest of the SCC support logic consists of TTL/RS-232-C interface chips to allow the SCC to communicate with modems or other RS-232-C type devices.

The connectors used for the serial channels are the standard 26-pin insulation displacement connector (IDC) wired so that when crimped to a cable with a DB-25 on the other end, the RS-232-C signals will come out on the correct pins.

PIO Circuitry

The PAD has two Z80 PIO chips on it. One is used for an optional terminal or computer interface if the user wishes to communicate with the PAD in parallel rather than serial mode. It can be left off if parallel interface is not required. A jumper is provided to continue the interrupt daisy-chain if this chip is not installed. If it is being used, its output signals are fed thru a 74LS244 octal buffer to provide enough drive to run the signals through a relatively long cable. The connector used for the parallel interfaces are the standard 26-pin IDC connectors and are wired the same as most parallel ports using this connector.

The auxilliary PIO is used to generate and receive several signals used by the PAD board. The parts of the aux PIO not needed for on-board functions are fed to a standard 26-pin connector, so they are available to the user for whatever he wants. In addition to the lines mentioned above that are used by the SCC interface, three lines from the output are sent to the channel A HDLC interface so they optionally can be used to control the speed of a multi-speed modem, such as might be used on hf.

Real-Time Clock and Support

The real-time-clock chip used is the National Semiconductor MM58167. In addition to keeping the time of day, it can generate timed interrupts to the CPU, and it also has an alarm function, allowing it to wake up the rest of a system at a preset time. Power for the real-time-clock chip is obtained from the plus twelve-volt bus thru a zener diode regulated power supply, some switching diodes and a battery supply. This allows for more time to cleanly shut down the system and prevent the clock chip from losing its memory.

The clock chip is not part of the Zilog Z80 family, and it does not generate any interrupt vector when it interrupts the CPU, so an octal tri-state latch was added to the board to provide an interrupt vector when the clock generates an interrupt. This vector is loaded by the CPU, so it can be programmed to be just about anything that the CPU will recognize.

The clock chip's power-down interrupt* is an open-drain output that is fed to the aux parallel 26-pin connector, allowing this signal to be used by external logic to power up a system.

Standby RAM Logic

The standby RAM is a 6514 low-power CMOS static RAM (organized as a 1024-by-4-bit device) that is addressed as a series of input/output ports. The upper three address bits to the RAM chip come from a four-bit latch. This allows the use of all 1024 nibbles in the RAM chip. The RAM is automatically deselected on power down, preventing it from being glitched by power transitions. The RAM can be used to store the amateur call, speed on both serial channels, whether the terminal/computer is using a serial or parallel channel, and many other PAD attributes that should be saved during power down. Using a battery-backed-up RAM instead of a NOVRAM allows the RAM to be updated much easier and more often, in addition to simplifying the hardware design.

Software Considerations for the PAD

The first software running on the PAD is a Z80 monitor modified to run on the PAD. The software required to run the PAD on AX.25 level 2 is being developed right now, and should be available just about the same time as the hardware. This software is written to take advantage of the PAD's features including the potential for downloading the protocol program and also allowing multiple connections. This software is taking a little longer to develop primarily because it is being written for a TDL Z80 assembler rather than using a higher-level language.

The Zilog 8530 SCC has sixteen write and nine read registers internally for each of the two serial channels. They are addressed by first writing a pointer to the control register of the channel of interest, then the requested register can be accessed. These registers allow each serial channels attributes to be programmed, from whether the channel is to be synchronous or asynchronous to the type of flag character used. The registers have to be programmed in a certain order to obtain proper operation, especially when using the SCC in HDLC mode. For more information on the SCC and its internals, see the <u>AMRAD Newsletter</u> for January of 1983.

The National MM58167A clock chip has 24 internal registers available to the programmer. Eight of these registers are the counters that contain the time, another eight contain the "alarm" time, and the rest are used to control or read the various internal functions of the chip, including such things as the interrupt handling, individual counter resets, and the standby interrupt (power down alarm) control.

The parallel input/output chips are standard Z80A PIO's. They have four modes of operation for each of the two parallel channels per chip (except for port B not being able to run mode 2. The four modes are:

Mode 0. Output mode.
Mode 1. Input mode.
Mode 2. Bidirectional mode.
Mode 3. Control mode.

The terminal PIO is capable of operating in any of the four modes with no problems. The auxilliary PIO is used to control and monitor several of the internal PAD operations, along with having a few bits left over for such things as modem-speed control. The aux PIO has the outputs of the transmit timers going to it for example, so if port A of the aux. PIO was put in mode 3, it could monitor the timers and generate an interrupt whenever the timers output changes state. The PIOs do generate Z80 style interrupt vectors, so this could be accomplished very easily, allowing an elegant recovery from a time-out condition.

PAD Availability

The PAD will be available only as a bare board for a while, due to capital limitations. I will be able to offer some of the harder-to-find components on a sporadic basis for a while. Hopefully we will be able to support the board better in the near future. Anyone interested in this board should write to me at the address listed above.

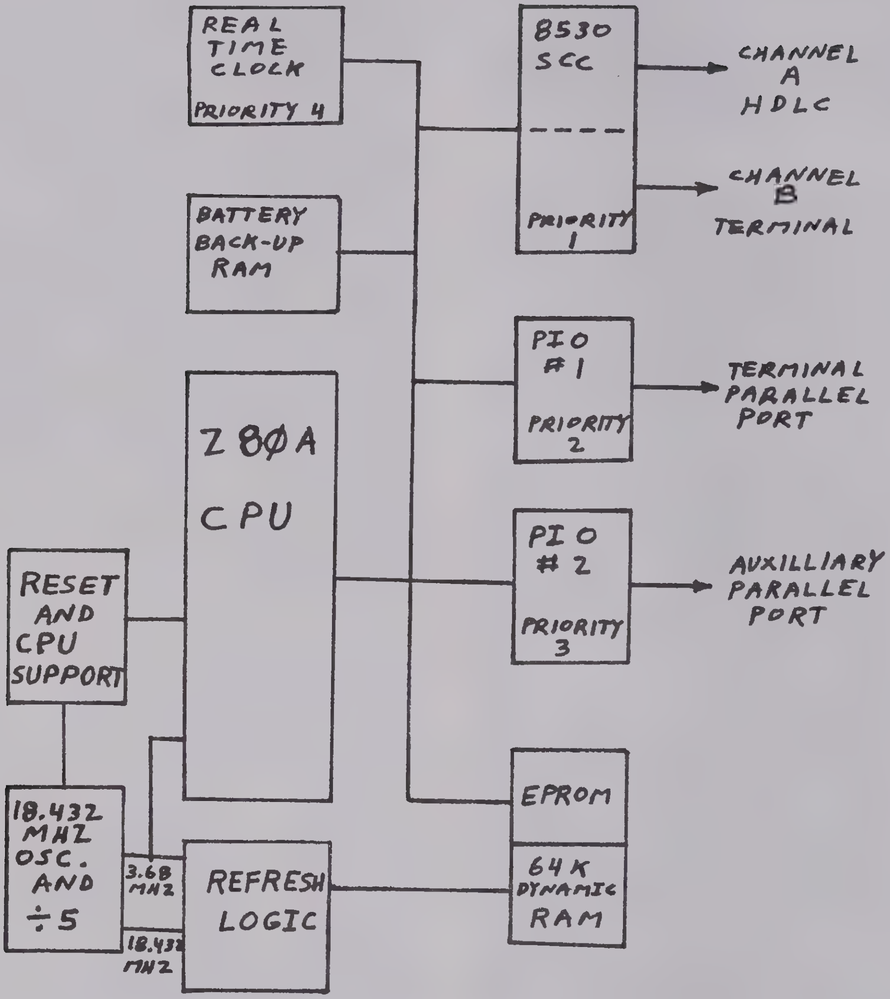


Fig. 1 - Block diagram of PAD.

DESIGN DECISIONS FOR THE TAPR TNC LINK LEVEL

David Henderson, KD4NL
2621 W. 164th St.
Torrance, CA 90504

Abstract

The decisions that were made up front on the software side of the TAPR project have had a very strong impact on the implementation and success of that project. Following is a review of some the design decisions that were made long before coding started, and a chronicle their impact upon implementation and performance.

Introduction

Before the software description starts, lets run through a quick overview of the hardware. The microprocessor is a Motorola 6809; the memory complement is 24 kilobytes of EPROM, 6 kilobytes of RAM, and 32 bytes of electrically erasable ROM. The peripheral chips consist of a Mostek 6551 asynchronous interface adapter, and a Western Digital 1933 HDLC interface chip. There is also an Mostek 6522 for clock support and onboard parallel I/O. The potential of using a parallel port for terminal I/O is provided with a Motorola 6820 parallel port chip. The computer system used as a software factory for the onboard software was an HP-64000 microcomputer development system present at the University of Arizona. This system made possible the installation of the TNC software on the TNC hardware.

The entire software development cycle was focused on X.25. The AT&T BX.25 document was taken as a reference for LAPB (reference 1), and the AMSAT document (reference 2) was taken as a reference for the header construction. The transition tables in the BX.25 document were followed very closely for the connect/disconnect sequences, but the I-frame manipulation was implemented in other ways, as described in more detail later on.

Architecture

The number one design choice was to write as much of the TAPR code in Pascal as possible. Pascal was chosen because it is a widely available high level language, and the existence of sophisticated compile time options for debugging implied the Pascal checkout could be started either on a big timesharing system or a microcomputer Pascal system.

The choice which influenced the rest of the implementation the most was to have the high level Pascal code sit in a tight loop checking flags that are continually being set and reset by interrupt code written in assembly language. This design decision paved the way for implementation and checkout of the Pascal source without having to have the hardware actually running, and greatly simplified the design of the Pascal code since it did not have to deal with interrupts. There was also a complicating factor in that flags are continually being set and reset to schedule future actions in the Pascal code; that is, the main loop of code was a sequence of IF and CASE statements that check these important flags. The action taken by the code is not immediately apparent upon reading the code; you have to know the meaning of the flags being manipulated.

The next design hurdle was the buffering; how many buffers should there be and what mechanisms were to be used to move data from one buffer to another? The main task, being a TNC, needed only two buffers, one for the incoming data flow and one for the outgoing data flow, and it was thought at one time that only two buffers would be needed. Once digipeating and control functions were considered, it was clear that four buffers were needed - there had to be a way to shuttle HDLC input data to the HDLC output queue and a way of taking terminal input data for commands, acting upon the commands, then printing a response to the terminal. The software was then broken up into four distinct sections; each section moves data from one buffer to another, with the possible side effects such as parameter changes. One section moved data from the HDLC input ring to the terminal output ring, another moves data from the terminal input ring to the HDLC output ring, a third took data from the terminal input ring and produced status messages in the terminal output ring, and the digipeating process move HDLC frames from the HDLC input ring to the HDLC output ring.

The next choice was to have the software 'know' as little as possible about the half duplex nature of the radio link. Previous implementations such as Vancouver Area Digital Communications Group protocol (VADCG for short) had used the poll/final bit in messages to turn the link around and have the receiving side turn into the transmitting side. I did not fully comprehend the use of the poll/final bit in this manner, and this use of the poll/final bit certainly conflicted with the LAPB usage. The design decision chosen for this "when do I send" problem can be simply stated: Only acknowledge receipt of messages or send messages when the modem signal data carrier detect is not present, or the data carrier detect signal has dropped at least once since receipt of a message. This rule means that message acknowledgements will be generated once for every sequence of information frames, and that there is a break in the received traffic before any new messages are queued up for transmission.

Consequences of the architecture

The Pascal implementation on the HP-64000 development system was not a full ISO standard Pascal. The major difference dealt with character strings; the HP Pascal had a STRING data type whereas the ISO standard has only arrays of characters. Unfortunately when the HP system writers adopted the STRING data type, they threw out completely any compatibility with ISO standard programs - character strings longer than one byte could not be used. This problem was 'solved' by including all character strings into one area of memory in EPROM. All references to constant strings had to reference the name of the array containing the data in this read only data area.

The Pascal code was checked out on two different computer systems prior to being installed on the TAPR board. The systems were a 36 bit mainframe and an 8 bit micro. The 36 bit mainframe checkout used routines to loop back HDLC input and output together in software; this allowed the software to connect to itself and allowed the basic logic to be checked out; all of this checkout was greatly speeded up via the symbolic debugger on the mainframe. The 8 bit micro allowed on the air tests with VADCG boards, and was invaluable in shaking down more basic logic problems. Again the routines to interface to terminal and HDLC I/O had to be changed to reflect the routines existing in the 8 bit micro system, but this is a pretty easy task to accomplish. The result of this staged checkout was a very robust implementation of X.25. There were bugs in the Pascal code, but they were only evident under extreme conditions.

Implementation of the architecture

There are four streams of data flow: Two directions for HDLC I/O and two directions for terminal I/O. Knowing when there is data present in the input streams and when there is enough room in the buffers for more characters in the output streams is handled by global variables. These variables are generally set by interrupt routines and reset when low level routines called by Pascal manipulate data associated by the flags. Each data stream has its own peculiarities, and the flag checking/clearing activity associated with each peculiarity will be covered.

The first stream of data is the HDLC input stream. Here a global variable exists which always contains the HDLC input top frame size, and is zero if there are no HDLC input frames placed into the HDLC input buffer and not yet processed by the Pascal program. The portion of the Pascal code that handles HDLC input notices that the variable is nonzero, and calls a low level routine which will move the HDLC data from the input ring buffer into a private Pascal buffer for further examination.

For HDLC output, there is a global variable which contains the number of free bytes in the HDLC output ring buffer. This cell is checked before any HDLC output frame is generated, and if there is not enough room in the HDLC output ring buffer for the potential output frame, then the generation of the output frame is deferred by simply not clearing the flags that cause the output frame to be generated. In the case of digipeated frames, if there is not enough room in the HDLC output ring buffer for the digipeated frame, then the digipeated frame is simply forgotten.

Terminal output is similar to HDLC output, but there are differences. There is a global variable which contains the number of bytes currently available in the output ring buffer. The routine called to queue up terminal data will always wait for space available in the output ring buffer to queue the character. The purpose in having the variable output ring free byte count is to avoid waiting for buffer space when I-frames come in on the HDLC input port. When an I-frame comes in that has a data portion too big to buffer, the data in the frame are ignored and the HDLC message "RNR" is scheduled for future transmission. This async output routine is freely callable from anywhere within the Pascal code, and does get called as a part of parameter displays, hex dumps, and internal debug routines. It was decided that when any of these activities were going on, no one would care if the Pascal code was spinning while waiting for more room in the async output buffer.

Now we come to the most interesting of the buffering problems, terminal input. The nature of X.25 is that there can be up to seven complete I-frames in flight at once. From the tight RAM limitations, it was clear that the input data for these I-frames could not be duplicated anyplace else in memory. The solution chosen this time around was to build a table describing the active portion of the terminal input ring buffer. The table is eight entries long, and each entry consists of a data start pointer into the terminal input ring buffer, and a data length from that start point. The table is eight entries long because that is the modulus for the X.25

sequence numbers, and these X.25 sequence numbers serve as indexes into the table. Part of the routine activity is to check to see if a new I-frame can be generated, and if so then a check is made for data to fill the I-frame. This checking is performed by calling a routine which returns a removal pointer and a size for data (if any) present within the terminal input ring. If there are data present, then the size will be nonzero, and another table entry can be constructed to describe an I-frame in flight. When I-frames get acknowledged the data space occupied in the terminal input ring buffer is marked no longer in use by advancing to the next sequence number and by changing a pointer, allowing reuse of the memory. One consequence of this design choice on input buffering is a "selective reject" (asking for fills), becomes a more difficult job to implement than if the other feasible approach of using linked lists had been made (It should be noted that selective reject is not part of X.25).

The next area that was simplified by the design decisions is the generation of HDLC frames for information transfer. There is a definite priority in X.25 for the kinds of frames that have to be sent. The priority scheme is implemented by the order in which flags are checked. These flags are generally set in the HDLC input routine, but may be set by anyone to schedule HDLC output. The flags that are checked and the order in which they are checked are: The send RNR flag (set when terminal buffer space gets low) to send a RNR frame, the send REJ flag (set when an out of order frame is received) to send a REJ frame, the received RNR flag must be reset (set when an RNR frame is received) to send an I-frame, the send RR flag (set when I-frame received) to send an RR frame. This section of code is also where the half-duplex decision must be made. The code to generate these output frames is only executed if the modem signal DCD (data carrier detect) is low or if the DCD signal dropped after any of the flags scheduling RNR, REJ or RR frames were set. This simple test is all it takes to prevent the sending of an RR frame for every I-frame received. Notice also that with the order in which the flags are checked, sending an I-frame will be attempted before sending an RR frame, so that if both sides have I-frames to send, then there are no RR messages sent when an I-frame would also acknowledge receipt to the other side.

Another detail that was made quite easy by the "no interrupts in Pascal" decision was the handling of multiple software timers. Actually, there are only two timers, the beacon timer and X.25 timer t1, but they are handled exactly the same way. The generalized timer code is implemented via a Pascal structure; this structure contains an expiration time and a Boolean flag that indicates whether or not the timer is running. "Time" is used loosely, for the only way the Pascal code is aware of the passage of time is by looking at yet another global variable. The time global variable is incremented at a one per second rate by an assembly language interrupt routine. Whenever a timer needs to be started, its expiration time is set to the time global variable plus the number of seconds in the timer interval, and a Boolean flag is set within the timer structure to indicate the clock is active. The big loop of code that is continually checking flags now has to check the timers for expiration, and this is quite easily done by comparing the global variable time with the expiration time in the timer.

Debugging and checkout

A subset of ISO standard Pascal was selected which would work both with the HP-64000 compiler and the two systems that I had available for writing and checkout. The real reason for this subset decision was to allow as much checkout of the software as possible in a friendly environment. The "no interrupt code in Pascal" decision made possible the replacement of low level routines in assembly language with routines written in Pascal which could invoke standard Pascal I/O. On the mainframe, a dummy set of HDLC input and output routines was written to loop back the HDLC output internally (from the Pascal programs point of view) to the HDLC input. On the 8 bit micro system, existing routines for HDLC input and output were modified to use new calling sequences and the Pascal code was actually executed on the air. In both debugging testbeds, the clock was simulated by incrementing the global variable holding the second tick whenever the code in the main loop noticed the system time of day change from a previous value. These sets of routines allowed checkout of the major logic flow of the Pascal software under a symbolic debugger. This self-test arrangement was extremely valuable, and allowed about ninety percent of the bugs in the Pascal code to be eliminated under the friendly environment of the symbolic debugger. Not everything could be checked, and the things that could not be checked were not setting flags for the low level routines (which didn't exist) or missing transitions in the state/event table that were never exercised during this self-test. One clear fallout of the debugging procedure was transportability, because the software was running on two different Pascal systems before the TAPR TNC software even met the TAPR TNC hardware.

Summary

The broad design decisions that were made before implementation of the TAPR TNC software served as an aid in implementation, supplying a framework that would support the detailed coding process. These decisions were made in the light of previous experience (and mistakes) in the implementation of three other systems similar to X.25. There were other choices that could have been made to supply the implementation framework, but my intent was to illustrate the decisions which made the TAPR TNC software almost write itself.

References

1. BX.25 Technical Reference, Issue 2, June 1980. American Telephone and Telegraph Company.

2. Protocol Specification for Level 2(link level) Version 1.1, Paul Rinaldo, W4RI, et al. October 10, 1982.

Unique Features of the TAPR TNC

by Lyle Johnson, WA7GXD
c/o Tucson Amateur Packet Radio Corp
P.O. Box 22888
Tucson AZ 85734

Background

The Tucson Amateur Packet Radio (TAPR) Terminal Node Controller (TNC) began as a local project done by a handful of Tucson-area Amateurs late in 1981. The project attracted enough attention to cause the formation of a formal club as well as an enlarged number of participants. As interest continued to grow, TAPR incorporated as a non-profit R&D group, and the TNC project changed from a local effort to include active participation in the design, implementation and testing phases with Amateurs from the West Coast to the Northeast.

The original Alpha-level TNC, of which only a dozen kits were distributed, led to the development of the current Beta-level TNC, now undergoing testing in dozens of sites, with site sizes ranging from three to over twenty stations. The total Beta distribution is approximately 175 preassembled TNCs.

While there have been, and continue to be, many other methods of getting involved with Packet activity, the TAPR TNC offers a number of unique features that are worth noting. It is the intent of the author that this paper accent those features, and it is the hope of TAPR that other experimenters may benefit from the experience and insight gained by the TAPR effort.

System Design

The TNC was designed as part of a system to allow an individual Amateur station to operate using the Packet mode with a minimum of integration problems. To this end, the required radio and terminal/computer interfaces are as general as possible. All I/O areas of the physical PC board surround a user wire-wrap area to allow for custom configuration. Further, the wire-wrap section has all significant power supply rails bordering one side.

Since it was designed as a system component, a radio-oriented MODEM (modulator-demodulator) is included on-board. To minimize the software design effort that might otherwise be required, an on-board microcomputer was designed to handle all aspects of lower-layered protocol implementation, with special emphasis in hardware to allow operation in an Amateur radio environment.

In any microprocessor- based device, software is a very important consideration. Consistent with the TAPR philosophy of modular flexibility, considerable efforts were made in hardware design to accomodate the desires of the software groups. Parallel software efforts were made, with the Pascal-based implementation of the "standard" protocol adopted in Washington on October 10, 1982, being initially distributed with the Beta TNCs. A FORTH- based design, implementing a dynamic addressing level-two protocol, is in process of being integrated on the TNC for the Beta testing phase.

Without proper documentation, most equipment is either rendered useless, or requires an inordinate amount of operator interaction to discover its "secrets." TAPR determined early on to provide as complete a manual set as was practical.

Finally, since it is generally conceded that Packet radio is in its infancy, the TNC design group made particular efforts to ensure flexibility as well as capacity for expansion.

MODEM Design

The MODEM incorporated in the TNC is designed for compatibility with the de facto standard Bell 202 tone pair, 1200 and 2200 Hz, using simple AFSK at a maximum data rate of 1200 baud.

The tone generator is straightforward, using the Exar 2206 FSK IC in a low-distortion, sine-wave configuration. The output is buffered by a simple +1 op- amp circuit, then ac- coupled to the radio interface connector. The output amplitude is user settable via a twenty- turn trimpot over a range from millivolts to volts.

During early experiments with the TNC, it was discovered that the WD 1933 HDLC controller did not allow direct control of the output state of its Tx Data pin when used in the NRZI mode. This complicated the issue of the US-required CW identification. In the end, use was made of the Exar 2206's analog multiplier input to allow for tone on-off keying of the CW ID. This input is buffered by a TTL inverter, allowing the tone state to be set under software control, independent of any other MODEM parameter.

The tone generator frequencies are easily changed via adjustment of individual twenty-turn trimpots. To facilitate such adjustment, the TNC hardware includes, and the present software supports, a frequency counter. This is a programmable 16-bit counter/timer used by the calibration routines as a period measurement timer. The software is designed such that the user may calibrate almost any frequency within the range of 10 Hz to over 230 kHz. The resolution is +/- 1.08 uSecs (one cycle of the system master clock of 921.6 kHz).

Although not strictly a MODEM function, control of the transmit control line is a necessary feature of the radio transmitter interface. To help prevent a "glitch" from allowing the TNC to lock up a radio frequency by continual assertion of the transmit control line, a monostable multivibrator, or "one-shot", is in series with the software- controlled transmit command line. A time constant of about 14-seconds allows a multiple- frame Packet transmission at the "normal" VHF packet rate of 1200 baud, along with a CWID of a typical Amateur call sign at 20 WPM. The particular circuit implemented utilizes a 555 timer IC in a non- retriggerable configuration. Non- retriggerable simply means that the 14- second timeout can't be held "on" by a continuous input -- the input must be removed before the 555 will accept another trigger. Thus, a Packet "channel" is protected from a runaway transmitter with minimal impact on system performance.

On the receive side, an Exar 2211 phase-locked loop (PLL) FSK demodulator is used. This circuit has a wide dynamic range input circuit (3 mV to 3-volts), and features both data and "carrier detect" outputs. (Note that the carrier referred to is the audio subcarrier, not the rf carrier.)

The circuit parameters have been optimized for operation at a data rate of 1200-baud, utilizing simple FSK with tones of 1200 Hz and 2200 Hz. After initial design, extensive testing was conducted and the circuit values "tweaked" for best operation. A few problems cropped up that may be of interest to others experimenting with similar MODEM designs.

The XR2211 chip is very sensitive to amplitude differences of the two input frquencies greater than about 3 db. At 6 db, the PLL will take so long to lock as to render the channel inoperative at the desired baud rate. Specifically, TAPR found that a typical 2-meter FM transceiver would not operate reliably above about 450 baud with the XR2211 used in an uncompensated circuit, while 1200 baud was easily realized in a compensated one. Various means of compensation were tried, and the best appeared to be an active filter. A CMOS switched capacitor filter was implemented, with a DIP header carrying the resistor network needed to configure the filter for a particular radio. The DIP header allows easy reconfiguration. This design is presented in detail in another paper in these proceedings.

In addition, a means of operator feedback for receiver audio level setting was deemed desireable, to prevent overloading of the filter while maintaining sufficient audio to prevent serious degradation of the s/n ratio. This was implemented with a simple back- to- back LED pair between the audio input buffer (a -10 amplifier) and the filter input network. Initial results from the field indicate this method to be both easy to use and effective.

Microcomputer

The TNC uses a 6809- based controller for logic implementation. This is believed to be the first Amateur radio application of the 6809 apart from individual efforts. The architecture of this microprocessor (uP) lends itself readily to high level, block-structured programming environments, and the first cut of TNC software is in fact written in Pascal.

The memory subsystem of the TNC utilizes a six- site bank of 28- pin sockets configured in the JEDEC "two line control" bytewide standard. This means that RAM, ROM, EPROM and/or EEPROM may occupy any socket. In addition, the memory may occupy any integral multiple of 2k- bytes capacity. To allow for future memory devices of greater density than used in the current TNC configuration, the memory map was carefully planned and executed in a bipolar Shottky PROM decoder.

The initial TNC consists of 6k- bytes of RAM contiguous from address 0, 4k-bytes of I/O space starting from address 2000H, and 24k- bytes of EPROM contiguous from address 0A000H through 0FFFFH. 2k-byte RAMs are used, along with 8k- byte EPROMs. 8k- byte RAMs can easily be accomodated, and 16k- byte and larger EPROMs can be used with a simple one- pin jumper (the socket wiring is presently compatible with 2764 and smaller EPROMs -- the one- pin change will make them compatible with 2764 and larger EPROMs). One Beta tester is building a piggyback adapter to utilize all 64k- bytes of address space.

Another unique aspect of memory design is the incorporation of the Xicor NOVRAM. This is a 256-bit device that can be accessed as a normal RAM (access time 300 nSec) as well as a 5- volt only EEPROM. This allows the user to store such parameters as station call sign, serial port parameters, HDLC port parameters, etc., and have the information retained after power is removed. However, the user can also change these parameters interactively. This provides a significant degree of freedom for the operator / experimenter, and is believed to be a first in Amateur radio.

Software Considerations

The Beta TNC is designed to support a variety of protocols by virtue of its large address space and third- generation uP. The initial Beta release software, discussed in much greater detail elsewhere in these proceedings, was designed to be compatible with the existing "Vancouver" protocol as well as implementing the lower layers of the AMRAD sponsored "AX.25" protocol adopted by the major US Packet groups in Washington, DC, on October 10, 1982.

The software package produced not only supports these modes, but it also supports operation of the TNC as a digipeater under either of the two protocols as well as a beacon (under either protocol). The TNC can in fact be used as all three of the above devices simultaneously, the only limit being that all functions must be under either one or the other of the two supported protocols, but not both.

Further, the software package supports certain powerful trace and debugging modes, which have proven to be of extreme value during system troubleshooting.

Other Features

The TNC design provides for software configuration of nearly every parameter associated with the I/O ports. For the serial (RS-232) channel, these parameters include baud rate, number of stop bits, parity options and data word length. For the HDLC port, the baud rate is under software control, using a 16-bit programmable timer. This allows using nonstandard data rates, such as the 400 baud being considered for experiments on the upcoming Phase 3B AMICON channel.

The parallel I/O port, to be supported in an upcoming revision to the Beta test software, is a full handshaking parallel interface, with separate 8-bit channels for input and output. Further, an EPROM programming adapter is being readied to allow users to bootstrap themselves along in software development and distribution.

Documentation

The TNC Beta document is a very comprehensive manual, both of the TAPR TNC in particular and Packet radio operation in general.

Over 140 pages in length, it includes chapters on setting up, operation and calibration, as well as in- depth discussions of the hardware, protocol and system design. It sets a standard seldom seen in manuals accompanying very expensive Amateur equipment, and far exceeds what many have come to deplore as "manuals" for the more common Amateur radio devices.

The Beta Test manual is currently in a state of rapid expansion as reports come in from Beta Test sites. The eventual manual will include expanded appendices with detailed information on interconnection with a plethora of commonly used 2-meter FM radios, as well as hardware and software installation data for many personal computers and terminals. The intent is to provide sufficient detail to allow a non-technical Amateur to successfully bring up a Packet radio station without other assistance.

Beta Test

Perhaps the most unique feature of the TAPR TNC is the manner in which it is being tested. Many innovative Amateur devices have been built as a result of a club project, involving perhaps 20 or 30 units. Commercial interests often build prototype units and allow selected customers to test them, who then offer feedback for the final production release of the product in question.

When TAPR decided to go ahead with the TNC design and implement it, the mailbox was full of requests from Amateurs desiring to be included in the initial distribution. Realizing the tremendous resource represented by these eager participants, and recognizing that the local Tucson "core" would be hard- pressed to fully shakedown, test and debug the TNC in a reasonable time frame, the decision was taken to allow a fairly large number of TAPR members the opportunity to contribute to the TNC design effort by providing a test bed.

The participants were informed that this was to be a test in both word and deed, and that all Beta testers were expected to file formal reports via Beta Test Coordinators to be located in each area. Further, to make the test sites as autonomous as possible, each local site was to determine its own coordinator. A network was established that now spans the USA and reaches, via AMSAT- sponsored participants, to Asia, Africa, Europe and Oceania.

Beta sites were to include both technical and non-technical Amateurs, and were to be self-suffcient as to support, modification, repair and updating of the TNCs within the site. It was felt that this would both relieve the pressures on the Tucson group and provide a reasonable cross-section of Amateurs with regard to technical expertise, climatic variation, rfi and frequency congestion (jammers and the like).

Further, by using this method, it was believed that a maximum number of Amateurs in varied locations would be exposed to the raucaus noises generated by the packet frames, and inquiries would lead to further interest in the mode. A local group could then "spread the packet gospel" and help ensure the further growth of the mode, even while it undergoes changes associated with its infancy.

Finally, many people who contacted us, and continue to contact us, indicated that there seemed to be no cohesive force amongst packeteers, nor central clearing point for information. TAPR has therefore attempted to fill this void, and the results to date have been most gratifying. By making Beta Test a national endeavor, participants are able to become involved in a broad- based, grassroots packet effort. By providing support for the TNC, people are reassured that, in the event they face real problems, there is a source for technical assistance, both in hardware and software.

The Beta Testing of the TAPR TNC is still in its initial phase, that of distribution. As of this writing, over 110 of the scheduled 170 TNCs have been shipped to a number of sites scattered across America. Most sites have had little trouble getting the TNCs on-the- air. Many radios, terminals and personal computers have been successfully interfaced.

Already, defect reports are coming in. A few significant hardware bugs have been noted, and work has commenced on solving the problems. Software- related problems have been reported, and the second release of Beta software will be appearing around the time of this conference.

It is believed by the author that this level of testing and cooperation has not been seen in Amateur radio before. The Beta Test participants have been very enthusiastic, as well as patient and supportive of the entire effort.

Conclusion

The TAPR TNC includes a variety of unique and innovative circuits, concepts and ideas. It is designed for both the technically inclined experimenter, for whom it offers a host of features accenting flexibility and expandability, as well as for the less- technically inclined operator, for whom it offers ease of operation.

The documentation of the TNC is at a level uncommon in Amateur radio, with a depth and breadth seldom seen outside of professional circles.

The initial software release incorporates many advanced features to allow the experimenter great latitude in optimizing system parameters, as well as flexibility in his particular system configuration. Compatibility with both the "Vancouver" and the "AX.25" protocols is provided.

The method of Beta Test is unique within Amateur radio endeavors, and is designed both the enhance the TNCs suitability for Packet radio use as well as increase the exposure of the general Amateur community to the strengths and benefits of this mode.

Acknowledgments

The author wishes to express his gratitude and thanks to the more than 250 members of TAPR, without whose support this project would never have gotten off the ground; to AMRAD and the ARRL for the opportunity to present this paper; to the Los Angeles and St. Louis contingents of TAPR for help in the areas of software, board layout and transformer procurement; and a special thanks from TAPR to Modular Mining Systems of Tucson for the extended use of their lab and other facilities, without which the TNC might never have been initially designed, nor built for so low a cost.

Modulation and Access Techniques for PACSAT

Phil R. Karn, KA9Q/2

Assistant Vice President for Engineering, Systems Analysis

Radio Amateur Satellite Corporation

ABSTRACT

This paper describes work underway within AMSAT to define modulation, channel access methods, and related system-level considerations in the design of the store-and-forward packet radio satellite known as PACSAT.

This is not intended as a comprehensive design specification, primarily because one doesn't yet exist! In particular, only those decisions primarily concerning spacecraft hardware design are emphasized here, since the details of control algorithms, protocols, etc, will reside in software capable of being changed and reloaded into the onboard computer(s) after launch.

1. Orbital Considerations

PACSAT is intended to operate in a low altitude, polar orbit with the special characteristic that the satellite is accessible from any point on earth at least twice every day, at about the same local time each day. These so-called "sun synchronous" orbits are frequently used by weather and earth resources missions. Oscars 6, 7, 8 and 9 are all in sun synchronous orbits, so many amateurs are already familiar with them.

The low altitude and relatively high velocity of such a satellite has several implications for communications:

1. For most earth locations, a sequence of two or three passes occurs twice daily, as the turning earth carries the station through the orbital plane every twelve hours.

2. Coverage at any given time is relatively small and continuously changing. A geostationary satellite "sees" a fixed portion (41%) of the earth's surface. By comparison, Oscar-8 covers all points on earth at least twice per day, but when it is directly over the north central US, it can only see North America.

3. Passes are short. A typical pass may last for only 15 minutes from horizon to horizon. For digital communications, the highest possible bit rate is desirable to maximize the amount of communication that can be carried during a pass, although for operational flexibility the spacecraft downlink transmission rate will be under the control of a command station.

4. Path losses are modest, approximately 25-30 dB lower than to geostationary satellites. This may allow the use of lower transmitter power, omnidirectional antennas, or less efficient modulation techniques, all of which help reduce costs.

5. Doppler shift is significant. At 70 cm, horizon-to-horizon Doppler shift can be as much as plus or minus 10 khz, requiring some form of automatic frequency tracking for optimum performance. Extra wide receiver filters and noncoherent demodulation will tolerate Doppler shift, but at the cost of reduced performance.

6. Propagation time is short, ranging from 3 milliseconds when the satellite is overhead to 12 milliseconds on the horizon. Therefore, a communication protocol requiring close interaction between the satellite and its users would not unduly penalize performance except for very small packets.

2. Downlink Margins

In designing a communications spacecraft, there are always practical limitations on power, size and weight, often with the emphasis on power. Hence, the power available for the spacecraft transmitter becomes the limiting factor in the overall system design. For this reason, it is helpful to start our analysis with a stated, realistic value. This immediately provides insight into the range of modulation and bit rate options available, therefore, AMSAT's best estimate of RF output power for PACSAT is 1-2 watts.

For a spacecraft in an Oscar-8 orbit (altitude 900 km), path loss on two meters would vary from 135 dB when the satellite is directly overhead to 147 dB with the satellite on the horizon, a range of 12 dB. Since even during an overhead pass the satellite spends most of its time "nearer" the horizon than directly overhead, we will be conservative and use the horizon figure in subsequent calculations.

Given that approximately 1 watt of transmitter output power is available on 2 meters, and that omnidirectional antennas are used both on the spacecraft and on the ground, receiver input power would be -147 dBW (-117 dBm, or .3 microvolts into 50 ohms.) A receiver with a bandwidth of 15 khz and a noise temperature of 300 K would generate an equivalent input noise power of -162 dBW, for a carrier-to-noise ratio of 15 dB, adequate for a low bit error rate with virtually any digital modulation scheme. However, since the satellite will move rapidly, use a real antenna with unavoidable pattern nulls, and often pass behind such real-world obstructions as trees and hills, additional margin is desirable.

This could be achieved in several ways, by

1. Reducing the receiver bandwidth. Each factor of two reduction would improve the carrier-to-noise ratio figure by 3 dB. However, for a given type of modulation, this reduces the bit rate, which is undesirable because it limits the amount of traffic that can be sent during the relatively brief passes.

2. Using more sensitive receivers. It is now quite easy to find inexpensive preamplifiers with low noise figures. However, external noise then becomes a factor, limiting the degree of improvement possible.

3. Using gain antennas with automatic tracking. Techniques for this have been experimented with by AMSAT members for several years, and soon will be within the realm of the average user of the Phase-3B spacecraft.

   Although the computation of antenna pointing angles has become very easy (the AMSAT ZX 81 project uses that very inexpensive personal computer for the task), the gain antennas are still relatively large and require fixed locations. We should therefore require gain antennas only as a last resort, in the sense that it should be possible to make effective use of PACSAT without them.

4. Using more efficient modulation methods. There exist techniques which can give much better performance in the presence of noise which are not yet widely used in the Amateur service. Many of these techniques have been widely used in commercial terrestrial and satellite services, but until now, they have often been considered out of reach of the average amateur because of cost and complexity. However, advances in digital electronics has de-coupled the issue of cost from complexity, bringing these techniques within reach of the average amateur.

3. Modulation Alternatives

We feel that the modulation method chosen is a crucial element of the PACSAT design, and I will spend the next part of this paper evaluating our alternatives.

3.1 AFSK-FM

Audio frequency frequency-shift-keying on an FM carrier is the technique currently in widespread use for terrestrial amateur packet radio, with the Bell 202 modem frequencies (1200 and 2200 hz) a de-facto standard. In the amateur satellite service, UOSAT-Oscar-9 uses bit-coherent AFSK-FM for its VHF and UHF telemetry, with tone frequencies of 1200 and 2400 Hz - close enough to those of the Bell 202 to allow the use of that (non-coherent and therefore non-optimum) modem by the majority of stations receiving telemetry.

AFSK-FM has several major advantages: it is cheap, simple, and allows the use of general-purpose transceivers without modification. Doppler tracking is relatively easy, since most amateur FM receivers have sufficient bandwidth to allow large frequency deviations (e.g, 1 khz) without significant degradation of bit error rate, and the modulation tone frequencies are not directly affected by Doppler shift.

Despite its simplicity, however, AFSK-FM has serious disadvantages for satellite use, which rule out its use in PACSAT:

1. Inefficient bandwidth utilization. A 15 khz channel is used by UOSAT to carry a (maximum) 1200 bps data stream, a bandwidth density of only .08 bits/hertz. Particularly

in the 2 meter band, spectrum efficiency is an important consideration.

2. Poor noise performance. Since AFSK-FM is essentially doubly-modulated FM, it exhibits a very sharp noise threshold at a relatively high carrier-to-noise ratio, and suffers greatly from impulse noise. Subjective experience with reception of the 350 milliwatt 145.825 MHz UOSAT-Oscar-9 telemetry beacon shows that pulse noise, e.g., from power lines, causes significant errors even at signal strengths otherwise sufficient to cause "full quieting" in the baseband FM channel. Local impulse noise and fades below threshold due to spacecraft rotation and polarization losses cause many errors, despite a theoretically good link margin.

3.2 FSK

Although in common use on the HF amateur bands, "straight" frequency shift keying (FSK or F1) has not yet come into widespread use on the higher frequency bands. FSK at VHF can be implemented with simple modifications to most conventional VHF-FM transceivers; a direct input to the modulator and a slicer on the discriminator output is required. The spectral efficiency of FSK can be as high as 1 bit/Hz; for our working bandwidth of 15 khz, FSK could realistically support a data rate of at least 10 kbps.

The Bell System's Advanced Mobile Phone Service (AMPS) uses NBFM voice transceivers with 8 khz peak deviation, 30 khz IF bandwidth, and discriminator detection to carry a biphase-encoded 10 kbps FSK signaling channel. In biphase encoding, the data stream is exclusive-ored with a clock at the bit rate, resulting in a signal with no DC component whose energy is concentrated about a frequency equal to the bit rate.

This allows an "indirect FM" (phase modulated) transmitter to be used, provided an integrator is inserted between the bi-phase encoder and the modulator. This is a direct adaptation of FM techniques used to encode data on disks and high-density magtapes. Since in FM the baseband (demodulator output) noise level increases with increasing frequency, a practical limit to the bit rate exists requiring relatively high receiver input C/N ratios when operating at high rates. In AMPS, considerable retransmission redundancy is also provided since multipath fading, not gaussian noise, is the primary source of errors in mobile radio.

AMPS is evidence that high rate FSK is practical given a sufficient C/N ratio. Performance would be better with PACSAT, since multipath fading is less severe in satellite channels than in terrestrial mobile radio, except when the satellite is near the horizon. For our working C/N figure of 15 dB, noncoherent reception of FSK (e.g., with an ordinary FM discriminator) would provide a theoretical bit error rate of about 10^-7; acceptable, but with little margin for implementation losses and fading. For example, if C/N dropped to 12 dB, the bit error rate would jump to 5 x 10^-4.

Because of the tight C/N margin, Doppler correction is required in order to allow narrow receiver bandwidths no wider than the signal. Since biphase encoding produces no baseband DC component, Doppler could be tracked by a simple integrator connected to the discriminator output.

We can rule out use of noncoherent FSK modulators for PACSAT, because it will be shown later that another modulation technique (MSK) exists which is compatible with simple FSK demodulation but also allows coherent detection with a 3.5 - 4 dB improvement in bit error rate. However, our analysis of noncoherent FSK is useful because it indicates the performance that could be expected if MSK is demodulated with an FM receiver.

3.3 DSPK

Differential Phase-Shift Keying (DPSK) is relatively new to amateur radio, but will be used by AMSAT for the Phase III spacecraft engineering beacon. DPSK has significant advantages: it is fairly bandwidth efficient, works very well in low carrier-to-noise levels, and can automatically track Doppler shift if correctly designed.

DPSK is actually a modified form of true PSK in that the *change* (or lack thereof) in carrier phase between each bit interval is used to determine the output state. In true PSK, the absolute phase of the carrier during each bit interval determines the output state, which requires an absolute phase reference at the receiver. If a clock is derived from the incoming data stream, there would be a 50-50 chance that the receiver would synchronize 180 degrees from the correct value, resulting in a 100% bit error rate! Differential PSK avoids this problem at the cost of having channel errors "propagate" through successive bits. However, in a packet environment where only a single bit error is needed to cause rejection of a packet and retransmission, extra errors "caused" by the first are of no consequence.

The noise performance of DPSK is considerably better than conventional FSK; for our 15 dB reference C/N, the bit error rate would decrease to 10^-10. However, the real advantage would be under marginal conditions: the bit error rate would not increase to 5 x 10^-4 until the C/N ratio had decreased to about 9 dB. Within a 15 kHz bandwidth DPSK could carry 15 kbps, although its noise performance would be degraded by such tight filtering; 9600 bps would be more realistic.

AMSAT has considerable experience with DPSK modulators and demodulators designed for 400 bps telemetry reception from Phase 3-B. Experiments have shown that non-linear transmitters are OK, and that 2.4 khz SSB transceivers are fine as long as compensation is made for the nonlinear phase response characteristic of SSB IF crystal filters.

The Phase 3 telemetry decoders use Costas loop carrier recovery which provides optimal performance, but may take an excessively long time to lock up in a multi-access packet environment. However, very simple DPSK demodulators exist that require no clock recovery circuit and are able to lock up in essentially a single bit time. These methods work at the expense of noise performance; the figures quoted above refer to this form of demodulation, while the Phase 3 demodulators do somewhat better.

NASA has also made extensive use of low cost, low-speed, (100-400 bps) doppler-tracking PSK systems with low altitude satellites for applications including remote data collection and search-and-rescue.

3.4 MSK

Minimum Shift Keying (MSK) is a hybrid of FSK and PSK. It can be regarded either as coherent FSK with a shift of exactly one-half of the data bit rate, or as PSK where the modulating waveform is a triangular ramp produced by integrating the binary input signal. Another equivalent way to look at MSK is as quadrature PSK (PSK with four possible phases instead of two) in which the quadrature channel carries the same data as the main channel but delayed by one-half bit period. In fact, the usual method for optimally decoding MSK involves building two PSK demodulators with combining circuitry; clearly this is more involved than a simple PSK demodulator.

One of MSK's advantages over PSK is that is requires minimal filtering to reduce its bandwidth to the minimum required. It has a constant envelope amplitude, unlike bandwidth limited PSK, allowing it to pass through a real-world linear spacecraft transponder with minimal intermodulation distortion to other signals. It can also be passed through a nonlinear (e.g., Class C) amplifier without the envelope distortion and resultant bandwidth-spreading that occurs with DPSK signals. The other advantage of MSK, perhaps the major one for our application, is that it can be decoded with simple noncoherent FM discriminators with a theoretical 3.6 dB loss of noise performance.

Optimally decoded MSK and PSK have almost the same performance in the presence of Gaussian noise. However, MSK has a significant advantage over PSK in cases of adjacent channel interference, due to MSK's smaller bandwidth. Tighter IF filters can be used with less performance degradation, and it should be easier to attain a rate of 15 kbps in our 15 khz bandwidth. Differential decoders eliminating the need for carrier recovery, similar to those mentioned earlier for DPSK, exist for MSK but are not as simple.

3.5 Discussion: MSK vs. DPSK

The "votes" are not yet all in among the PACSAT system definition and design team, although we have narrowed the choice to one between DPSK and MSK. MSK's most significant advantage is clearly its compatibility with simple noncoherent demodulators such as an FM discriminator. However, this penalizes those who want to "do it right", as optimal demodulation of MSK essentially requires building a PSK demodulator twice.

We could go with a form of DPSK essentially similar to that used by the Engineering Beacon on the Phase 3 spacecraft, except at a higher bit rate. While there is strong interest in MSK for AMICON (Phase 3) data communications, the relatively short time available to settle major hardware-related PACSAT issues could cause us to choose the simpler technology, i.e., DPSK. While the FSK demodulator compatibility feature is no doubt attractive, optimal demodulators for PACSAT would be produced by AMSAT on a relatively large scale, and would probably be a small fraction ($50 - $100) of the total station cost.

Either method would provide means for Doppler correction. The Phase-3B telemetry receivers use Costas loop demodulators for the DPSK signal, generating as a byproduct a correction voltage indicating the offset of the downlink carrier. This correction voltage can be taken out of the receiver and applied with the appropriate amount of gain to the transmitter, tuning its frequency to compensate for uplink Doppler. While the uplink channel demodulators will probably be able to track out the frequency shift without correction, we feel that minimizing channel lockup time is important enough that correction should be provided.

4. Access Conflict Resolution

PACSAT will be a multi-access satellite, intended to serve a number of users simultaneously attempting to send messages to the satellite. The downlink transmitter will be connected to the onboard computer, not directly to the uplink receiver as in conventional "bent-pipe" satellite transponders. Despite the short propagation delay, users will not be able to monitor the immediate status of an uplink channel by listening to the downlink, as it may be busy sending down a message intended for another user. Therefore, provisions must be made to resolve uplink access conflicts. (Naturally, since only one transmitter, the satellite, transmits on the downlink, access resolution is relevant only for the uplink.)

Assume for the moment that the satellite traffic will be "balanced", that is, the amount of traffic successfully received at the satellite will be approximately equal to the amount of traffic sent back to the ground when averaged over a sufficiently long period of time. It is agreed that this is an unlikely situation, which would only be true if PACSAT were to be used exclusively for point-to-point communications. Repeated transmission of the same information from the satellite (e.g., broadcast bulletins or spacecraft telemetry) would disproportionately increase downlink loading. However, it is my assertion that a balanced traffic assumption is a useful one, as it represents a "worst case" for system design.

All known methods which resolve contention between multiple uplink transmitters require overhead, and hence more bandwidth, than downlink transmissions for which there is only a single transmitter. We are therefore tentatively planning to use the 70 cm band, which has a 3-Megahertz Amateur Satellite Service allocation (435-438 MHz), for uplink transmissions to PACSAT and the smaller 2 meter band segment for the downlink.

In the following sections, I will describe two possible access methods for PACSAT, and compare their relative merits.

4.1 Pure Aloha With Multiple Uplink Channels

The Aloha method calls for each station to transmit at will, without concern to interference with other transmitters. (Since stations communicating with a satellite are usually far enough apart to prevent them from hearing each other, not much is gained by listening on the uplink frequency.) The well-known maximum theoretical throughput of an Aloha channel, above which delay time rises without bound, is 18%.

A very simple and attractive scheme therefore appears. If it is desired to balance uplink and downlink capacity, six uplink channels (6 x 18% = 108%) could be provided. Each one is "equal" to the others and scanned rapidly enough by the spacecraft's onboard computer to allow simultaneous reception, at least for a time, on all six. A user station would select one of the six uplink channels essentially at random whenever it has traffic for the satellite. Since the channels are all equivalent, all that matters is that the stations distribute their traffic across the channels in order to level out loading. This could be accomplished simply by allowing each station to choose an uplink frequency at random, changing it as often as desired, perhaps with each transmission. It can be shown that with a sufficient number of stations, traffic will tend to become evenly distributed over the channels.

It should be pointed out that to provide flow control, a requirement independent of the access method chosen, the spacecraft and ground computers will follow a synchronized "handshaking" protocol once a traffic transfer starts. If the ground computers are "patient" enough, that is, they allow enough time for processing and queuing delays aboard the satellite, collisions would result only when new stations initially access the satellite.

In addition to providing flow control, the go-ahead messages to each station could include a "recommended" uplink channel to use. Based on channel loading statistics kept in the spacecraft computer, the ground station would still be free to use any channel it wished, although following the recommendation would improve uplink traffic distribution.

4.2 Reservation Aloha

The "anarchy" of the Aloha system could be reduced somewhat, with an associated improvement in spectrum efficiency, at the cost of extra discipline in the ground station computers and added delay.

One of the uplink channels is designated as the "calling channel", on which stations transmit their initial requests for service to the satellite. This is in contrast to the last scheme, in which a new station may request service at any time on any channel. Requests would indicate the amount of service desired, and because they would be short, traffic on the calling channel would hopefully be well below the 18% "total bedlam" figure. The satellite responds by granting the requesting station permission to transmit its traffic on a specific frequency during a given time "slot". Depending on the length of the time slots and the tightness of their scheduling, each station might to compute and compensate for propagation delays which change continuously during a pass.

There are two advantages of Reservation Aloha:

1. New stations requesting service would not interfere with data exchanges already in progress on the working channels.

2. Due to the tight scheduling of the working channels, fewer of them might be necessary, reducing spacecraft hardware complexity.

4.3 Discussion: Pure Aloha vs. Reservation

These two schemes represent specific points in what is actually a fairly continuous spectrum of alternatives between "total anarchy" and "total discipline". The "more disciplined" reservation scheme with the designated calling channel can potentially provide better channel utilization than the pure Aloha method; however, it suffers from an "Achilles Heel" in that it is much more susceptible to jamming, accidental or otherwise, particularly on the calling channel. With any channel usable both for calling and working, the multi-channel Aloha system provides built-in redundancy against certain hardware failures as well as jamming. For this reason, along with the strongly attractive feature of simplicity, we feel that each channel should be equivalent, although by ground software convention one channel could be used primarily for initial service requests.

While the throughput of Aloha may seem low, the 18% figure is valid only for a very large number of users; "excess capacity" exists in systems with a small number of users, especially those in which one user presents most of the traffic load. If it turns out that uplink loading becomes a limiting factor (unlikely for reasons discussed earlier), it would be possible to change operations to a "slotted Aloha" access method. This would involve programming the ground station computers to "agree" that a reference event, e.g, the beginning of a certain telemetry frame periodically interspersed into the downlink data stream, represents the beginning of a packet slot. If the ground stations were to time their transmissions to coincide with such slots, the utilization of each channel could double to 37%, and this improvement could be obtained with no changes to spacecraft hardware or software. However, each station would have to compute and correct for the varying propagation delays to the satellite as in the Reservation Aloha system.

5. Summary

This paper has presented and discussed the various modulation and access method alternatives available to the PACSAT design team. It must be emphasized that the conclusions reached here are preliminary; only after considerable simulation, experimentation, and breadboarding activity will the final decisions be made.

In any case, it is probably true that we have already "over-engineered" the PACSAT uplink in that the downlink will almost certainly become the throughput-limiting factor. Now if we only had a few more watts of power....

6. Credits

The ideas presented here actually represent those of a large number of AMSAT people in addition to the author, including but not limited to:

- Dr. Thomas A. Clark, W3IWI, AMSAT USA President and perhaps the initial "instigator" of the PACSAT concept;

- Mr. Den Connors, KD2S/7, Assistant Vice President for Engineering, Spacecraft Systems and PACSAT Project Manager;

- Dr. John L. DuBois, W1HDX, Phase 3 Ground Command Station Coordinator;

- Mr. Jan A. King, W3GEY, Vice President for Engineering;

- Dr. Karl Meinzer, DJ4ZC, President AMSAT-DL;

- Dr. Stephen E. Robinson, W2FPY, Assistant Vice President for Engineering R&D

REFERENCES

[1] *Bell System Technical Journal*, January 1979. (Special issue on the Advanced Mobile Phone Service)

[2] *Reference Data for Radio Engineers*, Howard Sams & Co, Inc.

[3] J. L. Pearce, *Measured Performance Comparison of Fast FSK [MSK] and BPSK*, Canadian Electrical

Engineering Journal, Vol. 2 No. 4, 1977.

[4] W. B. Bruene, *MSK Simplified*, Collins Radio Company (released paper.)

[5] Mischa Schwartz, *Computer Communication Network Design and Analysis*, Prentice Hall, 1977.

[6] Dr. John L. DuBois, W1HDX. (AMSAT Phase 3 PSK notes).

[7] Bhargava, Haccoun, Matyas and Muspl, *Digital Communications by Satellite*, Wiley, 1981.

A BLOCK ORIENTED INTERFACE FOR CP/M* AND THE VADCG TERMINAL NODE CONTROLLER

Douglas Lockhart, VE7APU/3
29 Shamokin Drive
Don Mills, Ontario M3A 3H7
416-441-2417

Abstract

This paper describes a system of hardware and software which provides for the transfer of blocks of data between a VADCG Terminal Node Controller (TNC) and a CP/M system with a serial interface. Both the software to run in the TNC and in the CP/M system is included. The system provides block transfers, data transparency, flow control and error checking and retransmission in both directions over the interface.

Introduction

The software to implement the Link level of protocol for the VADCG Terminal Node Controller was developed in 1978. It is now in general use both in the U.S. and Canada and has even been implemented on other Terminal Node Controller boards. It has proven to be satisfactory for the purposes intended but many people recognize the need to implement the next higher level of protocol - the Packet or Network level protocol.

There have been a large number of proposals as to the form this protocol should take and I have made my own proposals in a paper published in the last Amateur Radio Computer Networking Conference. In spite of a large supply of proposals there is a distinct shortage of implementations. Part of the reason for this has been because of the need for some kind of consensus in the Amateur Radio fraternity. Notwithstanding this important concern, there is another reason why we don't have our Network level protocol implemented - it is a lot of work to get it going.

'What are the problems in implementing the Network level protocol?', you may ask. Well, unlike the Link level protocol which only had to be implemented to run in a TNC, parts of the Network level protocol have to be implemented to run in each microcomputer connected to the network. Furthermore, the TIP programs in the TNCs will have to be rewritten and some changes in the LIP programs are needed as well. In addition, the Network level protocol is much more complex than the link level protocol. I think one of the main stumbling blocks is the need to implement the protocol on two separate systems before any testing can be done.

Despite the above difficulties, I have begun the process of implementing the protocol and have broken the job down into steps that can be implemented and tested and then proceed to the next step. To alleviate the problem of having to make two implementations for different systems, I am only making one implementation for my CP/M system which I will hopefully be able to transport to another local Packeteer's CP/M system for testing. In order to make this program as transportable as possible to other CP/M systems I am only using the 8080 instruction set.

The programs here are not really any part of a higher level protocol but the function they perform will be needed by any higher level protocol that is adopted. The microcomputer program called 'PACKET' is basically a set of drivers for the serial interface between the microcomputer and the TNC. The program implementing the higher level of protocol in the microcomputer is called the Transmission Control Program or TCP. The TCP will use these drivers to transfer blocks of data that it has prepared to the TNC and it will also receive blocks of data from the TNC using these drivers.

The TCP is called upon by the programs running in the microcomputer to send data and receive data to and from various points in the network. In order to do this job, the TCP adds a header onto the outgoing blocks of data and because the bits and bytes in this header have a meaning based on their position in the block of data, there must be a mechanism to show where a block starts and ends in the serial data streams being passed accross the interface between the computer and the TNC. This mechanism, was lacking in all the TIPs that I had access to. Also, since flexibility in the setting of these bits was needed and any kind of restriction on the data being sent across the interface was undesirable, there had to be a mechanism for data transparency. This mechanism, too, was missing in all the TIPs that I had access to. Also, since data was being sent both ways at high speed by microprocessors, there had to be a mechanism for flow control in both directions across the interface. Also, since my serial interface used long RS-232 cables in a noisy environment, I occasionally got bit errors in the data especially at the higher speeds so I needed to have error detection in this interface. In some environments, error detection may not be necessary but I decided to play it safe and include it. Finally, error detection is not of much use unless you can correct the errors so I have incorporated a retransmission mechanism.

To summarize - the interface provides the following:

1. Block recognition.
2. Data transparency.
3. Flow control (in both directions)
4. Error detection.
5. Error correction.

A block has the following format:

SYN	DLE	STX	DATA	DLE	ETX	CRC	PAD
16H	10H	02H		10H	03H		FFH

The combination DLE-STX (ASCII Data Link Escape and Start of Text) indicates the start of a transparent block of data and the combination DLE-ETX indicates the end of the transparent block. To provide for data transparency a 'byte stuffing' technique is used - any time transparent data occurs that looks like a DLE, then an extra DLE is stuffed into the data stream. Therefore, the two byte combination DLE-DLE represents only a single data byte of 10H.

Flow control is accomplished using some hardware features of the TNC and the serial interface on the microcomputer. The RTS (Request to Send) and CTS (Clear to Send) lines are cross connected and controlled by the programs. When the output line is high it means 'You can send data to me now'. When the output line is low it means 'Don't send any data to me now.'

Error detection is accomplished using the two-byte CRC (Cyclic Redundancy Check) characters following the ETX character in the block. I am using the following polynomial to generate the CRC bytes:

$$X^{16} + X^{15} + X^{2} + 1$$

This is the usual polynomial used for synchronous protocols such as IBM BISYNC but is not the one suggested by the CCITT. On transmit, the CRC calculation is done on all transmitted characters after the STX and up to and including the ETX character. The stuffed bytes are included in the calculation and after the STX is processed, two bytes of zeroes are processed. On receive, the calculation is the same except that the two CRC bytes are used instead of the zero bytes and the result of the CRC calculation will then come out to zero if everything was received correctly.

The error correction mechanism employed also utilizes some of the hardware features of the TNC and the microcomputer. The DTR (Data Terminal Ready) and the DSR (DataSet Ready) lines are cross connected between the TNC and microcomputer. Whenever one side receives a block correctly, it reverses the state of its output line. If the other side does not detect the transition then, after a timeout, it retransmits the block.

Hardware Requirements

In order to use the program called 'TIPTTC' which runs in the VADCG TNC, you will need a VADCG TNC with the serial interface installed and an RS232 cable with wires going to the following pins installed (2,3,4,5,6,7 and 20).

In order to use the program called 'PACKET' which runs in a CP/M system, you will need to have a serial interface capable of handling 8-bit characters, direct software control of two lines of RS-232 levels, and the ability to read two input RS-232 lines with the software. Most CP/M systems have this capability. It is true that I could have written this software to only require the data lines (and I may yet do this) and the software would be slightly more transportable but more complicated and a little less efficient. The flow control and acknowledgment systems work very well because the software in the TNC is alerted by the interrupt system almost instantly when there is any change in level of the interface lines.

Software Requirements

The 'TIPTTC' program should interface with any of the common LIP programs being used with the VADCG board. I can only think of one thing to watch out for - the program uses variables in the CCA (Common Communications Area) from displacement 40H to 54H so you should check your LIP's usage of these areas and relocate them if your LIP uses part of the same area. Also, make sure your stack does not get extended down as low as displacement 54H in the CCA. This is a 'vanilla' TIP and in addition to the features described above, it only has provision for connect and disconnect. If you use this TIP you will have to do without those special functions you previously had. The other alternative is to add the functions to this program yourself. If you take this latter option I would very much like to hear from you as well as anyone else who uses these programs. I like to get 'feedback.'

The 'PACKET' program only needs a CP/M system with the aforementioned hardware features and some configuration modifications described in the next section.

Configuration Requirements

A. TIPTTC

A.1 At label 'BAUDRAT' the Baud rate may have to be changed. I am using 4800 Baud. In general it is best to have the rate as high as is reliable and convenient and should be 1200 or greater. However, lower Baud rates than 1200 would work as well.

A.2 At label 'ACKTO' there is a number which is related to the amount of time the TNC waits before retransmitting the block if no acknowledgment is received. This value has not been optimized from the first trial value. It is very non-critical and the value I chose for my system seems to work very well. It is probably quite a bit slower than required. You may experiment with different values.

A.3 At label 'RIMBUF' change the call sign to your own and if it is less than 6 characters, pad it on the right with blanks. Also, use upper case characters.

A.4 At label 'TERMNO' change your node number to whatever you want.

B. PACKET

B.1 In the section headed 'HARDWARE PORT EQUATES' you will have to change the port addresses to match the ports on your system.
B.2 In the sections headed 'CONTROL PORT BIT MEANINGS' and 'STATUS BIT MEANINGS' you will have to change the equates to match your system.
B.3 At label 'UARTINIT' change the code to initialize your serial interface UART to operate with 8 data bits and no parity bit. Also make the output lines going to pins 4 and 20 on the TNC are low. (The assumption here is that the jumper plug on the TNC is wired straight across)
B.4 At label 'SETRTS' change the code so that it makes pin 4 on the TNC end of the cable high.
B.5 At label 'CLEARRTS' change the code so that it makes pin 4 on the TNC end of the cable low.
B.6 At label 'FLIPDTR' make sure the code reverses the level on pin 20 of the TNC.
B.7 At label 'TESTTBE' test if data can be sent out to the UART and return non-zero status if it can.
B.8 At label 'TESTRDA' test if data is available from the UART and return non-zero status if it is.
B.9 At label 'TESTCTS' test the level of pin 5 coming from the TNC and return non-zero status if it is high.
B.10 At label 'TESTDSR' test the level of pin 6 coming from the TNC and compare it to the last tested level. If the value has changed, return non-zero status.
B.11 In routine 'KEYTEST' change the code to look for a character to be entered on your keyboard and if there is none, then go to 'OUTTEST'. It will probably have to be changed because my keyboard uses inverted logic.

Operation

Although the importance of the 'PACKET' program lies in the features provided by the drivers in it, I have added 25 instructions which allow the program to provide an immediately useful function. It will allow the user to use the keyboard and screen display in the CP/M system as if it were a terminal connected directly to the TNC. Because of the power of the driver code, it is a relatively trivial matter to add this function. Similarily, a program to transfer a file from the system or to the system is very easy to implement using the drivers.

To use the program as a terminal simulator, simply type in a line of data on the keyboard, the line will be sent in a block to the TNC when the line feed key is pressed. While data is being entered after the first character, no blocks will be received from the TNC. While a block is being received from the TNC, the keyboard is not tested so a line that you enter will not be mixed with data coming from the TNC.

To connect, type the call sign in upper case (which must be padded with blanks on the right if it is not 6 characters long) followed by control-A and then hit line feed to send it to the TNC. To disconnect, type any 6 characters (except for line feed) followed by control-B and then hit line feed. Sorry for this kludge but it is only temporary as I am planning to completely change the connect-disconnect procedures when I write the Transmission Control Program which is the next step in implementation of the Packet level protocols.

Summary

I hope these programs help those who are working on the implementation of the higher level protocols for an Amateur Radio digital communications network. It seemed to me that a program with these features would have to be one of the first steps in any kind of implementation but so far I have not heard of one. Perhaps someone out there has already written one and I have duplicated his effort. If so, then we are not doing enough advertising about what we have done. That is why I have taken this effort to disseminate the program.

The program listings here represent programs that have actually been running successfully so any problems encountered in transporting them to another system should be associated with the different environment and not with defects in the code. I can supply the programs on standard SS-SD CP/M format diskettes if necessary. Please enclose $3.00 with a blank diskette or $8.00 without a diskette when making your request. You will find the listings for the two programs on the following pages.
* CP/M is a trade mark of Digital Research

```
                ***********************************************************
                **       VADCG PACKET LEVEL TNC DRIVER FOR CP/M          **
                **   BY DOUG LOCKHART, VE7APU          JANUARY, 1983     **
                ***********************************************************
                ; LAST CHANGED JANUARY 31, 1983

                ***********************************************************
                *  THIS PROGRAM CONTAINS THE DRIVERS TO EXCHANGE TRANS-  *
                *  PARENT BLOCKS OF DATA BETWEEN A CP/M OPERATING        *
                *  SYSTEM AND A VADCG TERMINAL NODE CONTROLLER USING A   *
                *  MATCHING PROGRAM.  IT USES THE REQUEST TO SEND (RTS)  *
                *  AND CLEAR TO SEND (CTS) LINES FOR FLOW CONTROL AND    *
                *  THE DATA SET READY (DSR) AND DATA TERMINAL READY      *
                *  (DTR) LINES FOR ACKNOWLEDGEMENTS.  ONLY DATA INFOR-   *
                *  MATION IS PASSED ON THE DATA LINES.  THE PROGRAM      *
                *  USES 'BYTE STUFFING' TO ACHIEVE DATA TRANSPARENCY     *
                *  AND USES A CRC-16 TO DETECT ERRORS.  IF THE TRANS-    *
                *  MITTED DATA IS NOT ACKNOWLEDGED BY A CHANGE IN LEVEL  *
                *  THEN THE BLOCK IS SENT AGAIN.                         *
                ***********************************************************
                ;       MISCELLANEOUS EQUATES
0005 =          BIOS    EQU     5
                ;
                ;       ASCII EQUATES
000D =          CR      EQU     0DH     ; CARRIAGE RETURN
000A =          LF      EQU     0AH     ; LINE FEED
0010 =          DLE     EQU     10H     ; DATA LINK ESCAPE
0002 =          STX     EQU     02H     ; START OF TEXT
0003 =          ETX     EQU     03H     ; END OF TEXT
0016 =          SYN     EQU     16H     ; SYNCHRONIZING CHARACTER
00FF =          PAD     EQU     0FFH    ; PAD CHARACTER
                ***********************************************************
                ;       HARDWARE PORT EQUATES
0001 =          DATA    EQU     01H             ; UART DATA PORT
0000 =          CONTROL EQU     00H             ; UART CONTROL PORT
0000 =          STATUS  EQU     00H             ; UART STATUS PORT
0002 =          KEYBD   EQU     02H             ; KEYBOARD DATA PORT
                ;
                ;       CONTROL PORT BIT MEANINGS
0001 =          DTR     EQU     01H             ; NOT DATA TERMINAL READY
0002 =          RTS     EQU     02H             ; NOT REQUEST TO SEND
0004 =          BRS0    EQU     04H             ; BAUD RATE SELECT
0008 =          BRS1    EQU     08H             ; BAUD RATE SELECT
0010 =          WLS1    EQU     10H             ; WORD LENGTH SELECT
0020 =          WLS2    EQU     20H             ; WORD LENGTH SELECT
0040 =          SBS     EQU     40H             ; STOP BIT SELECT
0080 =          PI      EQU     80H             ; PARITY INHIBIT
                ;
                ;       STATUS BIT MEANING
0001 =          RDA     EQU     01H             ; RECEIVE DATA AVAILABLE
0002 =          KSTB    EQU     02H             ; NOT KEYBOARD STROBE
0004 =          PE      EQU     04H             ; PARITY ERROR
0008 =          FE      EQU     08H             ; FRAMING ERROR
0010 =          OE      EQU     10H             ; OVERRUN ERROR
0020 =          DSR     EQU     20H             ; NOT DATA SET READY
0040 =          CTS     EQU     40H             ; NOT CLEAR TO SEND
0080 =          TBE     EQU     80H             ; TRANSMIT BUFFER EMPTY

                ***********************************************************
0100                    ORG     100H
0100 31C903             LXI     SP,STACK        ; INITIALIZE STACK
0103 CD4301             CALL    UARTINIT        ; INITIALIZE UART
0106 CD3501     OUTTEST:CALL    WRITESTAT       ; ANY DATA IN TBUF?
0109 CA2301             JZ      LINETEST        ; NO, TRY TO RECEIVE SOME
010C DB00       KEYTEST:IN      STATUS          ; ANY KEYBOARD DATA?
010E E602               ANI     KSTB
0110 C20601             JNZ     OUTTEST         ; NO, TEST FOR LINE DATA
0113 DB02               IN      KEYBD           ; GET DATA
0115 CD3A01             CALL    DISPLAY         ; DISPLAY IT
0118 CD3A05             CALL    WRITE           ; PUT IT INTO BUFFER
011B FE0A               CPI     LF              ; WAS IT A LINE FEED?
011D CC1005             CZ      TCLOSE          ; YES,SEND DATA IN BUFFER
```

```
0120 C30601             JMP     OUTTEST         ; GO FOR MORE DATA
                LINETEST:
0123 CD3505             CALL    READSTAT        ; DATA IN RECEIVE BUFFER?
0126 CC3A04             CZ      BLOCKRX         ; NO, TRY TO RECEIVE SOME
0129 CA0C01             JZ      KEYTEST         ; NO, TEST KEYBOARD ENTRY
012C CD1805             CALL    READ            ; GET DATA BYTE FROM RBUF
012F CD3A01             CALL    DISPLAY         ; AND DISPLAY IT
0132 C32301             JMP     LINETEST
                WRITESTAT:
0135 3A9B02             LDA     TBUFNUM         ; GET COUNT
0138 B7                 ORA     A               ; AND TEST IT
0139 C9                 RET

013A F5         DISPLAY:PUSH    PSW
013B 5F                 MOV     E,A
013C 0E02               MVI     C,2
013E CD0500             CALL    BIOS            ; DISPLAY DATA IN (E)
0141 F1                 POP     PSW
0142 C9                 RET                     ; RETURN TO CALLER
                ***********************************************************
                ;       BASIC UART DRIVER ROUTINES
                ;
                ;       INITIALZATION OF UART
                UARTINIT:
0143 3EB7               MVI     A,PI+WLS1+WLS2+BRS0+DTR+RTS     ; 8 DATA,
0145 D300               OUT     CONTROL ; NO PARITY, DTR AND RTS OFF
0147 329801             STA     CTRL            ; SAVE CONTROL INFO
014A DB01               IN      DATA            ; CLEAR ANY RESIDUAL DATA
014C DB00               IN      STATUS          ; SAVE INITIAL DSR STATUS
014E E620               ANI     DSR
0150 329701             STA     DSRSTAT
0153 C9                 RET                     ; RETURN TO CALLER

                ;       ENABLE RTS (MEANS DATA CAN BE RECEIVED)
0154 3A9801     SETRTS: LDA     CTRL            ; GET CONTROL INFORMATION
0157 E6FD               ANI     0FFH-RTS
0159 D300               OUT     CONTROL
015B 329801             STA     CTRL
015E C9                 RET

                ;       DISABLE RTS ( MEANS DO NOT SEND ME ANY DATA )
                CLEARRTS:
015F 3A9801             LDA     CTRL            ; GET CONTROL INFORMATION
0162 F602               ORI     RTS
0164 D300               OUT     CONTROL
0166 329801             STA     CTRL
0169 C9                 RET

                ;       REVERSE VALUE OF DTR (TO ACKNOWLEDGE BLOCK )
016A 3A9801     FLIPDTR:LDA     CTRL            ; GET CONTROL INFORMATION
016D EE01               XRI     DTR             ; FLIP DTR
016F D300               OUT     CONTROL
0171 329801             STA     CTRL    ; SAVE UART CONTROL INFORMATION
0174 C9                 RET                     ; RETURN TO CALLER

                ;       TEST VALUE OF TBE (TRANSMIT BUFFER EMPTY)
0175 DB00       TESTTBE:IN      STATUS
0177 E680               ANI     TBE
0179 C9                 RET

                ;       TEST IF RECIEVE DATA IS AVAILABLE
017A DB00       TESTRDA:IN      STATUS
017C E601               ANI     RDA
017E C9                 RET

                ;       TEST VALUE OF CLEAR TO SEND
                ;       NON-ZERO FLAG IF CTS, ZERO FLAG IF NO CTS
017F DB00       TESTCTS:IN      STATUS
0181 E640               ANI     CTS
0183 FE40               CPI     CTS             ; NOTE SENSE INVERTED
0185 C9                 RET
```

```
                 ;       TEST IF VALUE OF DATA SET READY HAS CHANGED
                 ;       NON-ZERO FLAG IF DSR HAS CHANGED, ZERO IF NOT
0186 E5          TESTDSR:PUSH    H                ; DO NOT CHANGE HL
0187 C5                  PUSH    B                ; OR BC
0188 219701              LXI     H,DSRSTAT        ; POINT AT OLD DSR STATUS
018B 46                  MOV     B,M
018C DB00                IN      STATUS
018E E620                ANI     DSR
0190 77                  MOV     M,A              ; SAVE NEW DSR STATUS
0191 B8                  CMP     B                ; COMPARE OLD AND NEW
0192 C1                  POP     B                ; RESTORE REGISTERS
0193 E1                  POP     H
0194 C9                  RET                      ; RETURN WITH FLAGS SET
                 **********************************************************
0195 0000        CRC:    DW      0                ; CRC CALCULATION AREA
0197 00          DSRSTAT:DB      0                ; DSR STATUS SAVE AREA
0198 00          CTRL:   DB      0       ; CONTROL PORT INFORMATION
00FA =           MAXNUM  EQU     250     ; MAXIMUM AMOUNT OF DATA ALLOWED
0199 9C01        RPOINT: DW      RBUF    ; NEXT POINT TO GET DATA IN RBUF
019B 00          RBUFNUM:DB      0       ; NUMBER OF BYTES IN RBUF
019C             RBUF:   DS      253              ; RECEIVE BUFFER
0299 9C02        TPOINT: DW      TBUF    ; NEXT POINT TO PUT DATA IN TBUF
029B 00          TBUFNUM:DB      0       ; NUMBER OF BYTES IN TBUF
029C             TBUF:   DS      253              ; TRANSMIT BUFFER
0399                     DS      30H              ; STACK AREA
03C9 =           STACK   EQU     $
                 **********************************************************
                 ;       SEND BYTE OF DATA OUT TO SERIAL PORT
                 ;       DATA PASSED IN ACCUMULATOR

                 SENDDATA:
03C9 CD5205              CALL    CALCCRC          ; INCLUDE IN CRC
03CC E5          SEND:   PUSH    H
03CD C5                  PUSH    B                ; SAVE B&C
03CE 4F          SEND1:  MOV     C,A              ; SAVE DATA TEMPORARILY
03CF 210100              LXI     H,1              ; DELAY FOR BUG IN UART
03D2 2B          SEND2:  DCX     H
03D3 7C                  MOV     A,H
03D4 B5                  ORA     L                ; IS IT 0?
03D5 C2D203              JNZ     SEND2
03D8 CD7501      SEND3:  CALL    TESTTBE          ; IS TBUF EMPTY?
03DB CAD803              JZ      SEND3   ; NO, KEEP LOOPING UNTIL IT IS.
03DE CD7F01      SEND4:  CALL    TESTCTS          ; IS CLEAR TO SEND UP?
03E1 CADE03              JZ      SEND4            ; NO, CAN'T SEND YET
03E4 79                  MOV     A,C              ; GET BACK DATA BYTE
03E5 D301                OUT     DATA
03E7 C1                  POP     B                ; RESTORE B
03E8 E1                  POP     H
03E9 C9                  RET                      ; RETURN TO CALLER

                 ;       SEND DATA IN TBUF TO THE UART TRANSPARENTLY

                 SENDTBUF:
03EA E5                  PUSH    H
03EB C5                  PUSH    B
03EC 219B02              LXI     H,TBUFNUM        ; POINT TO TBUF BYTE CNT
03EF 4E                  MOV     C,M              ; SAVE IN C
                 SENDTBUF1:
03F0 23                  INX     H                ; POINT TO NEXT BYTE
03F1 7E                  MOV     A,M              ; GET IT
03F2 CDC903              CALL    SENDDATA         ; SEND IT
03F5 FE10                CPI     DLE              ; WAS IT DLE?
03F7 CCC903              CZ      SENDDATA         ; IF SO, SEND IT AGAIN
03FA 0D                  DCR     C                ; DECREMENT COUNT
03FB C2F003              JNZ     SENDTBUF1        ; CONTINUE SENDING
03FE C1                  POP     B
03FF E1                  POP     H
0400 C9                  RET                      ; RETURN TO CALLER

                 ; SEND FORMATTED BLOCK TO UART
                 BLOCKTX:
0401 3E16                MVI     A,SYN
```

```
0403 CDCC03              CALL    SEND
0406 3E10                MVI     A,DLE
0408 CDCC03              CALL    SEND
040B 3E02                MVI     A,STX
040D CDCC03              CALL    SEND
0410 210000              LXI     H,0              ; INITIALIZE CRC AREA
0413 229501              SHLD    CRC
0416 CDEA03              CALL    SENDTBUF         ; SEND DATA IN TBUF
0419 3E10                MVI     A,DLE            ; THEN DLE-ETC SEQUENCE
041B CDC903              CALL    SENDDATA         ; INCLUDE IN CRC
041E 3E03                MVI     A,ETX
0420 CDC903              CALL    SENDDATA         ; INCLUDE IN CRC
0423 CDDE04              CALL    SENDCRC          ; FINALLY SEND CRC BYTES
0426 CDF204              CALL    CHECKRX          ; TRY TO RECEIVE
0429 CDD004              CALL    WAITDSR          ; WAIT FOR DSR TO CHANGE
042C CA0104              JZ      BLOCKTX ; DIDN'T CHANGE, SEND BLOCK AGAIN
042F AF                  XRA     A                ; A <-- 0
0430 329B02              STA     TBUFNUM          ; INDICATE TBUF IS EMPTY
0433 219C02              LXI     H,TBUF           ; POINT TO START OF TBUF
0436 229902              SHLD    TPOINT
0439 C9                  RET                      ; RETURN TO CALLER

                 ;       READ A FORMATTED TRANSPARENT BLOCK OF DATA
043A CD5401      BLOCKRX:CALL    SETRTS           ; ALLOW OTHER END TO SEND
                 BLOCKRX1:
043D CDAE04              CALL    RECEIVE          ; READ A BYTE FROM LINE
0440 C8                  RZ      ; RETURN WITH ZERO STATUS IF TIMED OUT
0441 FE10                CPI     DLE              ; IS IT DLE?
0443 C23D04              JNZ     BLOCKRX1         ; NO, KEEP TRYING
0446 CDAE04              CALL    RECEIVE          ; GOT DLE, TRY FOR STX
0449 C8                  RZ      ; RETURN WITH ZERO STATUS IF TIMED OUT
044A FE02                CPI     STX              ; IS IT STX?
044C C23D04              JNZ     BLOCKRX1         ; NO, TRY FOR DLE AGAIN
                 BLOCKRX2:                        ; ENTRY FROM BLOCKTX
044F 219C01              LXI     H,RBUF           ; POINT TO START OF RBUF
0452 229901              SHLD    RPOINT
0455 CD8004              CALL    RCVRBUF          ; RECEIVE DATA INTO RBUF
                                 ; UNTIL A CONTROL SEQUENCE IS RECEIVED
0458 C8                  RZ      ; RETURN ZERO STATUS IF LINE TIMES OUT
0459 FE03                CPI     ETX              ; WAS IT ETX?
045B C23D04              JNZ     BLOCKRX1         ; UNEXPECTED SEQUENCE
045E CDAE04              CALL    RECEIVE          ; RECEIVE FIRST CRC CHAR
0461 C8                  RZ                       ; RETURN HERE IF TIME OUT
0462 CDAE04              CALL    RECEIVE          ; RECEIVE SECOND CRC CHAR
0465 C8                  RZ                       ; RETURN HERE IF TIME OUT
0466 CD5F01              CALL    CLEARRTS         ; STOP OTHER END
0469 2A9501              LHLD    CRC              ; CHECK IF CRC WAS OK
046C 7C                  MOV     A,H
046D B5                  ORA     L
046E C27C04              JNZ     BLOCKRX3         ; NO GOOD
0471 219B01              LXI     H,RBUFNUM        ; SAVE DATA COUNT
0474 71                  MOV     M,C
0475 CD6A01              CALL    FLIPDTR          ; GOOD, REVERSE DTR LINE
                                                  ; TO ACKNOWLEDGE BLOCK
0478 3EFF                MVI     A,-1             ; RETURN NON-ZERO STATUS
047A B7                  ORA     A                ; BLOCK RECEIVED OK
047B C9                  RET                      ; RETURN TO CALLER
                 BLOCKRX3:
047C 3E00                MVI     A,0              ; RETURN WITH ZERO STATUS
047E B7                  ORA     A                ; NO BLOCK RECEIVED
047F C9                  RET


                 ;       RECEIVE DATA PORTION OF BLOCK, RETURNS WHEN A
                 ;       CONTROL SEQUENCE FOUND IN THE TRANSPARENT TEXT
0480 210000      RCVRBUF:LXI     H,0              ; INITIALIZE CRC TO 0
0483 229501              SHLD    CRC
0486 219C01              LXI     H,RBUF           ; POINT TO START OF RBUF
0489 0E00                MVI     C,0              ; BYTE COUNT = 0
                 RCVRBUF1:
048B CDAE04              CALL    RECEIVE          ; GET A BYTE FROM LINE
048E C8                  RZ      ; RETURN HERE WITH ZERO STATUS IF TIMEOUT
```

```
048F FE10          CPI     DLE              ; WAS IT DLE?
0491 CA9A04        JZ      RCVRBUF3         ; YES, LOOK AT NEXT BYTE
          RCVRBUF2:
0494 77            MOV     M,A              ; PUT INTO BUFFER
0495 23            INX     H                ; INCREMENT RBUF POINTER
0496 0C            INR     C                ; INCREMENT COUNT
0497 C38B04        JMP     RCVRBUF1         ; LOOP FOR NEXT BYTE
          RCVRBUF3:
049A CDAE04        CALL    RECEIVE
049D C8            RZ      ; ZERO STATUS RETURN IF LINE TIMES OUT
049E FE10          CPI     DLE      ; IS IT A TRANSPARENT DLE?
04A0 CA9404        JZ      RCVRBUF2         ; YES, GO PUT INTO BUFFER
04A3 C9            RET     ; RETURN WITH CONTROL BYTE IN ACCUMULATOR
                                            ; AND NON-ZERO STATUS

          ;        TRY TO READ FROM LINE WITH LONG TIMEOUT
04A4 E5   RECEIVL:PUSH     H
04A5 CD5401        CALL    SETRTS           ; ALLOW OTHER END TO SEND
04A8 21A00F        LXI     H,4000           ; LONG TIMEOUT VALUE
                           ; SHOULD BE ADJUSTED FOR BEST RESULTS
04AB C3B504        JMP     RECEIV1

          ;        TRY TO READ FROM LINE, IF LINE TIMES OUT,
          ;        RETURN WITH ZERO STATUS
04AE E5   RECEIVE:PUSH     H
04AF CD5401        CALL    SETRTS           ; ALLOW OTHER END TO SEND
04B2 21D007        LXI     H,2000           ; SHORT TIMEOUT VALUE,
                           ; ADJUST FOR ABOUT 2 CHAR TIMES
04B5 CD7A01 RECEIV1:CALL   TESTRDA          ; ANY RECEIVED DATA?
04B8 CAC204        JZ      RECEIV2          ; NO, DECREMENT TIME
04BB DB01          IN      DATA             ; GET DATA BYTE
04BD CD5205        CALL    CALCCRC          ; INCLUDE IT IN CRC
04C0 E1            POP     H
04C1 C9            RET     ; GOOD RETURN WITH NON ZERO STATUS
04C2 2B   RECEIV2:DCX      H                ; DECREMENT TIMER
04C3 7C            MOV     A,H
04C4 B5            ORA     L                ; IS TIME OVER
04C5 C2B504        JNZ     RECEIV1          ; NO, KEEP TRYING
04C8 CD5F01        CALL    CLEARRTS         ; OOPS, TIMED OUT,
                           ; DROP RTS SO OTHER SIDE WILL STOP
04CB 3E00          MVI     A,0      ; RETURN WITH ZERO STATUS
04CD B7            ORA     A
04CE E1            POP     H
04CF C9            RET

          ;        WAIT FOR DSR TO CHANGE USING TIMEOUT
          WAITDSR:
04D0 011027        LXI     B,10000          ; DELAY - ALTER AS REQ'D
          WAITDSR1:
04D3 CD8601        CALL    TESTDSR          ; CHECK FOR DSR CHANGE
04D6 C0            RNZ                      ; RETURN IF IT HAS
04D7 0B            DCX     B
04D8 78            MOV     A,B
04D9 B1            ORA     C                ; IS TIME OVER?
04DA C2D304        JNZ     WAITDSR1         ; NO CONTINUE TESTING
04DD C9            RET                      ; UNSUCCESSFUL RETURN

          ;        SEND THE CRC BYTES
04DE AF   SENDCRC:XRA      A                ; FINISH CRC CALCULATION
04DF CD5205        CALL    CALCCRC
04E2 CD5205        CALL    CALCCRC
04E5 3A9601        LDA     CRC+1            ; SEND FIRST CRC CHAR
04E8 CDCC03        CALL    SEND
04EB 3A9501        LDA     CRC              ; SEND SECOND CRC CHAR
04EE CDCC03        CALL    SEND
04F1 C9            RET

          ;        CHECK IF WE CAN RECEIVE A BLOCK NOW
          ;        THIS ROUTINE IS CALLED AFTER A BLOCK HAS BEEN
          ;        TRANSMITTED TOALLOW THE OTHER SIDE TO GET A
          ;        CHANCE TO SEND TO US
04F2 3A9B01 CHECKRX:LDA    RBUFNUM          ; IS THERE ANY DATA LEFT
04F5 B7            ORA     A                ; IN RECEIVE BUFFER?
04F6 C0            RNZ                      ; YES, CAN'T RECEIVE
04F7 CD5401        CALL    SETRTS           ; ENABLE RTS TO ALLOW
                                            ; OTHER SIDE TO SEND
          CHECKRX1:
04FA CDA404        CALL    RECEIVL          ; READ WITH LONG TIMEOUT
04FD C8            RZ                       ; TIMED OUT , RETURN
04FE FE10          CPI     DLE              ; IS IT A DLE?
0500 C2FA04        JNZ     CHECKRX1         ; NO, KEEP LOOKING
0503 CDA404        CALL    RECEIVL          ; NOW LOOK FOR A STX
0506 C8            RZ                       ; TIMED OUT SO RETURN
0507 FE02          CPI     STX              ; IS IT START OF TEXT?
0509 C2FA04        JNZ     CHECKRX1         ; NO, KEEP LOOKING
050C CD4F04        CALL    BLOCKRX2         ; NOW GO READ TRANSP.TEXT
050F C9            RET                      ; ZERO STATUS IF TIMEOUT
                                  ; NON-ZERO IF BLOCK WAS RECEIVED

          ;        SEND A BLOCK OF TRANSMIT DATA TO THE LINE IF
          ;        THERE IS ANY DATA IN THE BUFFER
0510 3A9B02 TCLOSE: LDA    TBUFNUM          ; GET COUNT IN BUFFER
0513 B7            ORA     A                ; TEST FOR DATA
0514 C40104        CNZ     BLOCKTX          ; SEND BLOCK IF ANY DATA
0517 C9            RET                      ; RETURN TO CALLER

0518 E5   READ:    PUSH    H                ; SAVE HL
0519 219B01 READ1: LXI     H,RBUFNUM        ; POINT AT COUNT IN RBUF
051C 7E            MOV     A,M
051D B7            ORA     A                ; IS THERE ANY LEFT?
051E C22A05        JNZ     READ2            ; YES
0521 CD1005        CALL    TCLOSE           ; SEND ANY DATA IN TBUF
0524 CD3A04        CALL    BLOCKRX          ; RECEIVE ANOTHER BLOCK
0527 C31905        JMP     READ1            ; TRY TO DO READ AGAIN
052A 35   READ2:   DCR     M                ; DECREMENT COUNT
052B 2A9901        LHLD    RPOINT           ; GET READ POINTER
052E 7E            MOV     A,M              ; GET DATA BYTE
052F 23            INX     H                ; INCREMENT POINTER
0530 229901        SHLD    RPOINT           ; AND SAVE AGAIN
0533 E1            POP     H                ; RESTORE HL
0534 C9            RET            ; RETURN TO CALLER WITH DATA IN A

          READSTAT:
0535 3A9B01        LDA     RBUFNUM ; GET COUNT OF DATA IN BUFFER
0538 B7            ORA     A                ; TEST IT
0539 C9            RET           ; NON-ZERO STATUS IF DATA PRESENT

053A F5   WRITE:   PUSH    PSW              ; SAVE DATA
053B E5            PUSH    H                ; SAVE HL
053C 2A9902        LHLD    TPOINT           ; GET POINTER INTO TBUF
053F 77            MOV     M,A              ; PUT DATA INTO BUFFER
0540 23            INX     H                ; INCREMENT POINTER
0541 229902        SHLD    TPOINT
0544 219B02        LXI     H,TBUFNUM        ; POINT TO COUNT IN TBUF
0547 7E            MOV     A,M              ; INCREMENT COUNT
0548 3C            INR     A
0549 77            MOV     M,A
054A FEFA          CPI     MAXNUM           ; IS BUFFER FULL?
054C CC0104        CZ      BLOCKTX          ; YES, SEND BLOCK NOW
054F E1            POP     H                ; RESTORE HL
0550 F1            POP     PSW              ; RESTORE DATA
0551 C9            RET

          ; CRC CALCULATION ROUTINE
          ; USES BYTE PASSED IN ACCUMULATOR TO INCLUDE IN CRC
          ; RESTORES ALL REGISTERS AND STATUS
0552 E5   CALCCRC:PUSH     H
0553 C5            PUSH    B
0554 F5            PUSH    PSW
0555 0608          MVI     B,8
0557 4F            MOV     C,A
0558 2A9501        LHLD    CRC
055B =    CALCCRC1:EQU     $
055B 79            MOV     A,C
```

```
055C 07                 RLC
055D 4F                 MOV     C,A
055E 7D                 MOV     A,L
055F 17                 RAL
0560 6F                 MOV     L,A
0561 7C                 MOV     A,H
0562 17                 RAL
0563 67                 MOV     H,A
0564 D26F05             JNC     CALCCRC2
0567 7C                 MOV     A,H
0568 EE80               XRI     80H
056A 67                 MOV     H,A
056B 7D                 MOV     A,L
056C EE05               XRI     05H
056E 6F                 MOV     L,A
             CALCCRC2:
056F 05                 DCR     B
0570 C25B05             JNZ     CALCCRC1
0573 229501             SHLD    CRC
0576 F1                 POP     PSW
0577 C1                 POP     B
0578 E1                 POP     H
0579 C9                 RET
             ;
057A                    END
```

```
                ; ************************************************************
                ; **     VADCG TERMINAL NODE COMMUNICATIONS PROGRAM         **
                ; **     BY DOUG LOCKHART, VE7APU       JANUARY, 1983       **
                ; ************************************************************
                ; LAST CHANGED: JANUARY 31, 1983

                ; TERMINAL INTERFACE PROGRAM FOR INTERFACING TO A CP/M
                ; SYSTEM.  THIS PROGRAM IS WRITTEN TO RUN IN THE VADCG
                ; TERMINAL NODE CONTROLLER.  IT INTERFACES WITH A LINK
                ; INTERFACE PROGRAM (LIP) RUNNING AT ADDRESS 0 IN MEMORY.
                ; THIS VERSION IS WRITTEN TO USE THE 8250 PROGRAMMABLE
                ; UART TO COMMUNICATE WITH A COMPUTER.
                ; THE BASIC FEATURES OF THIS TIP ARE:
                ; TRANSFER OF DATA IN BLOCKS
                ; RTS FLOW CONTROL FROM DIGITAL EQUIPMENT TO TIP
                ; AND CTS FLOW CONTROL FROM TIP TO DIGITAL EQUIPMENT
                ; ACKNOWLEDGEMENTS TO BLOCKS RECEIVED BY A CHANGE IN DTR
                ; ACKNOWLEDGEMENTS TO BLOCKS SENT BY A CHANGE IN DSR
                ; CRC-16 CHECKING OF ALL DATA BLOCKS
                ; ERROR RECOVERY BY RETRANSMISSION IF NO ACKNOWLEDGMENT
                ; USES BYTE STUFFING TECHNIQUE FOR DATA TRANSPARENCY

                INCTB   MACRO   ?D
                        IF      NOT NUL ?D
                        MVI     A,?D
                        ENDIF
                        RST     2
                        ENDM

                INCLB   MACRO   ?D
                        IF      NOT NUL ?D
                        MVI     A,?D
                        ENDIF
                        RST     3
                        ENDM

                COMPARE MACRO
                        RST     5
                        ENDM

                SIM     MACRO
                        DB      30H     ; SET INTERRUPT MASK
                        ENDM

                RIM     MACRO
                        DB      20H     ; READ INTERRUPT MASK
                        ENDM


                ; RAM CONSTANT - CHANGE FOR DIFFERENT RAM LOCATION
1000 =          LORAM   EQU     1000H   ; START OF RAM STORAGE


                ; NON-ZERO STATUS MEANS LINE BUFFER ADDRESS IS IN HL REG.
                ; ZERO STATUS MEANS NO BUFFER IS READY
                NEXTIN  MACRO
                        RST     4
                        ENDM


                ; 8255 PARALLEL I/O EQUATES

0008 =          PORTA   EQU     8       ; PORT A INPUT AND OUTPUT
0009 =          PORTB   EQU     9       ; PORT B INPUT AND OUTPUT
000A =          PORTC   EQU     0AH     ; PORT C INPUT AND OUTPUT
000B =          CONTROL EQU     0BH     ; CONTROL PORT OUTPUT ONLY


                ; BAUD RATE EQUATES
0004 =          BAUD384 EQU     4       ; DIVISOR FOR 38,400 BAUD
0008 =          BAUD192 EQU     8       ; DIVISOR FOR 19,200 BAUD
0010 =          BAUD96  EQU     16      ; DIVISOR FOR 9600 BAUD
```

```
0020 =          BAUD48  EQU     32        ; DIVISOR FOR 4800 BAUD
0040 =          BAUD24  EQU     64        ; DIVISOR FOR 2400 BAUD
0080 =          BAUD12  EQU     128       ; DIVISOR FOR 1200 BAUD
0100 =          BAUD600 EQU     256       ; DIVISOR FOR 600 BAUD
0200 =          BAUD300 EQU     512       ; DIVISOR FOR 300 BAUD
0400 =          BAUD150 EQU     1024      ; DIVISOR FOR 150 BAUD
0476 =          BAUD134 EQU     1142      ; DIVISOR FOR 134.5 BAUD
0573 =          BAUD110 EQU     1395      ; DIVISOR FOR 110 BAUD
0800 =          BAUD75  EQU     2048      ; DIVISOR FOR 75 BAUD
0C00 =          BAUD50  EQU     3072      ; DIVISOR FOR 50 BAUD


                ; 8250 SERIAL I/O EQUATES

                ; REGISTER EQUATES
0000 =          RBR     EQU     0         ; RECEIVE BUFFER REGISTER  (R)
0000 =          THR     EQU     0         ; TRANSMIT HOLDING REGISTER (W)
0001 =          IER     EQU     1         ; INTERRUPT ENABLE REGISTER  (W)
0002 =          IIR     EQU     2         ; INTERRUPT IDENT. REGISTER (R)
0003 =          LCR     EQU     3         ; LINE CONTROL REGISTER (R/W)
0004 =          MCR     EQU     4         ; MODEM CONTROL REGISTER (R/W)
0005 =          LSR     EQU     5         ; LINE STATUS REGISTER (R/W)
0006 =          MSR     EQU     6         ; MODEM STATUS REGISTER (R/W)
0000 =          DLL     EQU     0         ; DRIVER LATCH (LSB) (W)
0001 =          DLM     EQU     1         ; DRIVER LATCH (MSB) (W)

                ; INTERRUPT ENABLE EQUATES
0001 =          ERBFI   EQU     1         ; ENABLE RECEIVED DATA INTERRUPT
0002 =          ETBEI   EQU     2         ; ENABLE TRANSMITTER
0004 =          ELSI    EQU     4         ; RECEIVER LINE STATUS INTERRUPT
0008 =          EDSSI   EQU     8         ; ENABLE MODEM STATUS INTERRUPT

                ; INTERRUPT IDENTIFICATION EQUATES
0001 =          IPEND   EQU     1         ; '0' IF INTERRUPT PENDING
0002 =          IID0    EQU     2         ; INTERRUPT IDENTIFICATION BIT 0
0004 =          IID1    EQU     4         ; INTERRUPT IDENTIFICATION BIT 1

                ; LINE CONTROL EQUATES
0001 =          WLS0    EQU     1         ; WORD LENGTH SELECT BIT 0
0002 =          WLS1    EQU     2         ; WORD LENGTH SELECT BIT 1
0004 =          STB     EQU     4         ; STOP BIT SELECT
0008 =          PEN     EQU     8         ; PARITY ENABLE
0010 =          EPS     EQU     10H       ; EVEN PARITY SELECT
0020 =          SPTY    EQU     20H       ; STICK PARITY
0040 =          SBRK    EQU     40H       ; SET BREAK
0080 =          DLAB    EQU     80H       ; DRIVER LATCH ACCESS BIT

                ; MODEM CONTROL EQUATES
0001 =          DTR     EQU     1         ; DATA TERMINAL READY
0002 =          RTS     EQU     2         ; REQUEST TO SEND
0004 =          OUT1    EQU     4         ; OUT1 LINE ON 8250
0008 =          OUT2    EQU     8         ; OUT2 LINE ON 8250
0010 =          LOOP    EQU     10H       ; MODEM LOOP CONTROL BIT

                ; LINE STATUS EQUATES
0001 =          DR      EQU     1         ; DATA READY
0002 =          OE      EQU     2         ; OVERRUN ERROR
0004 =          PE      EQU     4         ; PARITY ERROR
0008 =          FE      EQU     8         ; FRAMING ERROR
0010 =          BI      EQU     10H       ; BREAK INTERRUPT
0020 =          THRE    EQU     20H       ; TRANSMITTER HOLDING REG EMPTY
0040 =          TSRE    EQU     40H       ; TRANSMITTER SHIFT REG EMPTY

                ; MODEM STATUS EQUATES
0001 =          DCTS    EQU     1         ; DELTA CLEAR TO SEND
0002 =          DDSR    EQU     2         ; DELTA DATA SET READY
0004 =          TERI    EQU     4         ; TRAILING EDGE RING INDICATOR
0008 =          DRLSD   EQU     8         ; DELTA RX LINE SIGNAL DETECT
0010 =          CTS     EQU     10H       ; CLEAR TO SEND
0020 =          DSR     EQU     20H       ; DATA SET READY
0040 =          RI      EQU     40H       ; RING INDICATE
```

```
0080 =          RLSD    EQU     80H       ; RECEIVE LINE SIGNAL DETECT

0017 =          RIMD    EQU     17H       ; REQUEST INITIALIZATION MODE
0008 =          MSE     EQU     08H       ; MASK SET ENABLE BIT


                ; COMMON COMMUNICATIONS AREA

                ; CIRCULAR TERMINAL BUFFER VARIABLES

1000 =          CCA     EQU     LORAM     ; COMMON COMMUNICATIONS AREA ADR.
1004 =          CTBIE   EQU     CCA+4     ; CURRENT TERMINAL BUF INP. ENTRY
1006 =          OTBE    EQU     CCA+6     ; OLDEST TERMINAL BUFFER ENTRY
1008 =          TBIP    EQU     CCA+8     ; TERMINAL BUFFER INPUT POINTER
100A =          TBOP    EQU     CCA+0AH   ; TERMINAL BUFFER OUTPUT POINTER
100C =          LTBOE   EQU     CCA+0CH   ; LAST TERMINAL BUF OUTPUT ENTRY
100E =          CTBOE   EQU     CCA+0EH   ; CURRENT TERMINAL BUF OUT ENTRY

                ; CIRCULAR LINE BUFFER VARIABLES

1012 =          LBPE    EQU     CCA+12H   ; LINE BUFFER PROCESSING ENTRY
1014 =          CLBE    EQU     CCA+14H   ; CURRENT LINE BUFFER ENTRY ADDR.
1016 =          OLBE    EQU     CCA+16H   ; OLDEST LINE BUFFER ENTRY
1018 =          LBIP    EQU     CCA+18H   ; LINE BUFFER INPUT POINTER
101A =          LBOP    EQU     CCA+1AH   ; LINE BUFFER OUTPUT POINTER

                ; MISCELLANEOUS

1000 =          STAT1   EQU     CCA       ; MAINLINE STATUS BYTE

                ; THE FOLLOWING VARIABLES ARE FOR EXCLUSIVE USE BY TIP

101C =          BUFCOUNT EQU    CCA+1CH   ; CURRENT INPUT BUFFER COUNT
101D =          OUTCOUNT EQU    CCA+1DH   ; CURRENT OUTPUT BYTES REMAINING
1040 =          WAIT    EQU     CCA+40H   ; CHARACTER DELAY VALUE
1042 =          MSRSAVE EQU     CCA+42H   ; LATEST MODEM STATUS REGISTER
1043 =          INTFLAG EQU     CCA+43H   ; INTERRUPT ROUTINE FLAGS
0001 =          RXBUSY  EQU     01H       ; RECEIVE INTRPT ROUTINE ACTIVE
0002 =          TXBUSY  EQU     02H       ; TRANSMIT INTRPT ROUTINE ACTIVE
1044 =          CRC     EQU     CCA+44H   ; CRC CALCULATION AREA
1046 =          RCRC2   EQU     CCA+46H   ; SECOND RECEIVED CRC BYTE
1047 =          RCRC1   EQU     CCA+47H   ; FIRST  RECEIVED CRC BYTE
1048 =          TCRC2   EQU     CCA+48H   ; SECOND TRANSMIT CRC BYTE
1049 =          TCRC1   EQU     CCA+49H   ; FIRST TRANSMIT CRC BYTE
104A =          RNEXT   EQU     CCA+4AH   ; CURRENT RECEIVE ROUTINE ADDRESS
104C =          TNEXT   EQU     CCA+4CH   ; TRANSMIT ROUTINE ADDRESS
104E =          RDISP   EQU     CCA+4EH   ; RECEIVE INTERRUPT ROUTINE ADDR.
1050 =          TDISP   EQU     CCA+50H   ; TRANSMIT INTERRUPT ROUTINE ADDR
1052 =          DFLAG   EQU     CCA+52H   ; DISPATCH FLAG
0001 =          CRCTX   EQU     01H       ; CRC ROUTINE IN USE BY TX DISP.

                ; ASCII EQUATES
000D =          CR      EQU     0DH       ; ASCII CARRIAGE RETURN
000A =          LF      EQU     0AH       ; ASCII LINE FEED
001B =          ESC     EQU     1BH       ; ASCII ESCAPE CHARACTER
0002 =          STX     EQU     02H       ; ASCII START OF TEXT
0003 =          ETX     EQU     03H       ; ASCII END OF TEXT
0010 =          DLE     EQU     10H       ; ASCII DATA LINK ESCAPE
0016 =          SYN     EQU     16H       ; ASCII SYNCHRONIZATION CHARACTER
00FF =          PAD     EQU     0FFH      ; TRAILING PAD CHARACTER

00FF =          TRUE    EQU     0FFH      ; FOR IF CONDITION TESTS
0000 =          FALSE   EQU     0         ; FOR IF CONDITION TESTS

                ; ***************************************************
                ; **              CONFIGURATION EQUATES            **
                ; **      VALUES CHANGE FOR EVERY CONFIGURATION    **
                ; ***************************************************
                ;


0003 =          FORMAT  EQU     WLS1+WLS0        ; UART FORMAT (8 DATA,
                                          ; NO PARITY)
0020 =          BAUDRAT EQU     BAUD48  ; CURRENT BAUD RATE

00FF =          CUSHION EQU     255     ; THE MINIMUM NUMBER OF BYTES
                                ; AVAILABLE IN THE TERMINAL BUFFER THAT
                                ; ARE REQUIRED BEFORE A RECEIVE
                                ; OPERATION IS STARTED.

2710 =          ACKTO   EQU     10000   ; ACKNOWLEDGE TIMEOUT COUNT
                                        ; (PRELIMINARY VALUE)

                ***********************************************************
0800                    ORG     800H    ; THIS PROGRAMS EPROM START ADR.

                ; ENTRY JUMP TABLE

0800 C31508             JMP     TIPINIT ; INITIALIZATION ENTRY POINT
0803 C34808             JMP     RST55   ; INTERRUPT FROM 8250
0806 C30608             JMP     $       ; UNUSED INTERRUPT ENTRY POINT
0809 C3100A             JMP     DISPRX  ; TO DISPATCHER ROUTINE
080C 0C17564537RIMBUF   DB      12,RIMD,'VE7APU'        ; CONNECT BUFFER
0814 C8         TERMNO  DB      200     ; THIS NODES TERMINAL NUMBER
                ***********************************************************
                TIPINIT:
                        ; SET BAUD RATE IN SERIAL PORT
0815 3E80               MVI     A,DLAB
0817 D303               OUT     LCR
0819 3E20               MVI     A,LOW BAUDRAT
081B D300               OUT     DLL             ; BAUD RATE DIVISOR LSB
081D 3E00               MVI     A,HIGH BAUDRAT
081F D301               OUT     DLM             ; BAUD RATE DIVISOR MSB

                        ; DEFINE CHARACTER FORMAT OF SERIAL DATA
0821 3E03               MVI     A,FORMAT
0823 D303               OUT     LCR     ; UPDATE LINE CONTROL REGISTER

                        ; UNMASK INTERRUPTS FROM SERIAL INTERFACE
                        RIM             ; GET CURRENT INTERRUPT MASK IN A
0825+20                 DB      20H     ; READ INTERRUPT MASK
0826 E606               ANI     00000110B       ; RESET RST5.5 MASK BIT
0828 F608               ORI     MSE     ; SET MASK SET ENABLE BIT
                        SIM             ; ENABLE RST5.5 INTERRUPTS
082A+30                 DB      30H     ; SET INTERRUPT MASK

                        ; CLEAR OUT RECEIVE BUFFER REGISTER
082B DB00               IN      RBR

                        ; SET UP INITIAL DISPATCH ROUTINES
082D 214309             LXI     H,EXIT  ; SET RECEIVE INTERRUPT TO IDLE
0830 224A10             SHLD    RNEXT
0833 21160B             LXI     H,WAITLIP       ; WAITING FOR LIP BLOCK
0836 225010             SHLD    TDISP
0839 21140A             LXI     H,WAITTB        ; WAITING FOR FREE CUSHION
083C 224E10             SHLD    RDISP
                ; ENABLE RECEIVED DATA AVAILABLE AND MODEM STATUS INTRPT
083F 3E09               MVI     A,ERBFI+EDSSI   ; RECEIVE AND MODEM
0841 D301               OUT     IER     ; UPDATE INTERRUPT REGISTER

                        ; BRING UP RLSD (OUT1 = RLSD)
0843 3E04               MVI     A,OUT1
0845 D304               OUT     MCR     ; UPDATE MODEM CONTROL REGISTER

                        ; RETURN TO LIP FOR COMPLETION OF INITIALIZATION
0847 C9                 RET
                ***********************************************************
0848 F5         RST55:  PUSH    PSW
0849 E5                 PUSH    H
084A D5                 PUSH    D
084B C5                 PUSH    B
084C DB02               IN      IIR     ; GET INTERRUPT IDENT INFORMATION
084E FE04               CPI     IID1    ; RECEIVED DATA AVAILABLE INTRPT?
0850 CA8A08             JZ      RXINT   ; GO TO RECEIVE INTERRUPT ROUTINE
```

```
0853 FE02               CPI     IID0    ; IS IT TRANSMIT BUFFER EMPTY
0855 CA4909             JZ      TXINT   ; GO TO TRANSMIT INTRPT ROUTINE
0858 B7                 ORA     A       ; MODEM STATUS INTERRUPT?
0859 CA5F08             JZ      MSINT   ; TO MODEM STATUS INTRPT ROUTINE
085C C34309             JMP     EXIT    ; UNKNOWN INTERRUPT, RETURN

085F DB06       MSINT:  IN      MSR     ; GET MODEM STATUS
0861 324210             STA     MSRSAVE ; SAVE MODEM STATUS FOR DISPATCH
0864 4F                 MOV     C,A     ; SAVE IT
0865 E601               ANI     DCTS    ; HAS CTS CHANGED?
0867 C46D08             CNZ     CTSINT  ; YES GO HANDLE CTS CHANGE
086A C34309             JMP     EXIT

086D 79         CTSINT: MOV     A,C     ; GET MODEM STATUS BACK
086E E610               ANI     CTS     ; TEST CTS BIT
0870 CA7608             JZ      DISABLETX       ; OFF, DISABLE TRANSMIT
0873 C37D08             JMP     ENABLETX        ; TRY TO ENABLE TRANSMIT

                DISABLETX:
0876 DB01               IN      IER     ; GET INTERRUPT ENABLE REGISTER
0878 E6FD               ANI     0FFH-ETBEI
087A D301               OUT     IER     ; TURN OFF TRANSMIT INTERRUPTS
087C C9                 RET

                ENABLETX:
087D 3A4310             LDA     INTFLAG ; IS TRANSMITTER BUSY?
0880 E602               ANI     TXBUSY
0882 C8                 RZ              ; NO, RETURN
0883 DB01               IN      IER     ; GET INTERRUPT ENABLE REGISTER
0885 F602               ORI     ETBEI
0887 D301               OUT     IER     ; ENABLE TRANSMIT INTERRUPTS
0889 C9                 RET
                ***********************************************************
088A DB00       RXINT:  IN      RBR     ; READ DATA FROM SERIAL PORT
088C 2A4A10             LHLD    RNEXT   ; GO TO ROUTINE ADDRESS IN RNEXT
088F E9                 PCHL

0890 FE10       RSTART: CPI     DLE     ; IS IT A DATA LINK ESCAPE?
0892 C24309             JNZ     EXIT    ; NO
0895 219E08             LXI     H,RSTX  ; YES, NOW WAIT FOR START OF TEXT
0898 224A10             SHLD    RNEXT
089B C34309             JMP     EXIT

089E FE02       RSTX:   CPI     STX     ; IS IT START OF TEXT
08A0 C2AC08             JNZ     RSTX1   ; NO
08A3 21B508             LXI     H,RDATA ; YES, HANDLE TRANSPARENT DATA
08A6 224A10             SHLD    RNEXT
08A9 C34309             JMP     EXIT
08AC 219008     RSTX1:  LXI     H,RSTART        ; FALSE START GO BACK
08AF 224A10             SHLD    RNEXT           ; TO BEGINNING
08B2 C34309             JMP     EXIT

08B5 FE10       RDATA:  CPI     DLE     ; IS IT A DLE?
08B7 CAC308             JZ      RDATA1  ; YES
08BA CDEE08             CALL    RPUT    ; NO, PUT DATA INTO BUFFER
08BD CA0609             JZ      RESTART ; ERROR, RESET BUFFER AND RESTART
08C0 C34309             JMP     EXIT            ; FROM BEGINNING
08C3 21CC08     RDATA1: LXI     H,RCONTROL      ; RECEIVE CONTROL
08C6 224A10             SHLD    RNEXT           ; CHARACTER NEXT
08C9 C34309             JMP     EXIT

                RCONTROL:
08CC FE10               CPI     DLE     ; IS IT A SECOND DLE?
08CE C2E008             JNZ     RCONTROL1       ; NO, CHECK FOR ETX
08D1 CDEE08             CALL    RPUT    ; YES, PUT DLE IN BUFFER
08D4 CA0609             JZ      RESTART ; ERROR, RESET BUFFER AND RESTART
08D7 21B508             LXI     H,RDATA ; GO BACK FOR MORE DATA
08DA 224A10             SHLD    RNEXT
08DD C34309             JMP     EXIT
                RCONTROL1:
08E0 FE03               CPI     ETX     ; IS IT END OF TEXT?
```

```
08E2 C20609             JNZ     RESTART ; NO, ERROR - RESTART
08E5 211D09             LXI     H,R1CRC ; NEXT RECEIVE FIRST CRC CHAR
08E8 224A10             SHLD    RNEXT
08EB C34309             JMP     EXIT

08EE 4F         RPUT:   MOV     C,A     ; SAVE DATA IN REGISTER C
08EF 2A0610             LHLD    OTBE    ; PUT DATA INTO BUFFER
08F2 EB                 XCHG
08F3 2A0810             LHLD    TBIP
                        INCTB   1
08F6+3E01               MVI     A,1
08F8+D7                 RST     2
08F9 C8                 RZ      ; RETURN WITH ZERO STATUS IF OVERFLOW
08FA 220810             SHLD    TBIP    ; UPDATE POINTER IF OK
08FD 71                 MOV     M,C     ; MOVE DATA INTO BUFFER
08FE 211C10             LXI     H,BUFCOUNT      ; INCREMENT COUNT OF DATA
0901 34                 INR     M
0902 7E                 MOV     A,M
0903 FEFB               CPI     251     ; HAVE WE GOT 251 BYTES NOW?
0905 C9                 RET             ; ZERO STATUS IF TOO MANY BYTES

0906 3E00       RESTART:MVI     A,0     ; SET COUNT IN BUFFER TO ZERO
0908 321C10             STA     BUFCOUNT
090B 2A0410             LHLD    CTBIE   ; SET INPUT POINTER JUST BEFORE
                        INCTB   1       ; DATA AREA
090E+3E01               MVI     A,1
0910+D7                 RST     2
0911 220810             SHLD    TBIP
0914 219008             LXI     H,RSTART        ; AND RESTART RECEIVER
0917 224A10             SHLD    RNEXT
091A C34309             JMP     EXIT

091D 324710     R1CRC:  STA     RCRC1   ; SAVE FIRST CRC CHARACTER
0920 212909             LXI     H,R2CRC ; NOW GET SECOND CRC CHARACTER
0923 224A10             SHLD    RNEXT
0926 C34309             JMP     EXIT

0929 324610     R2CRC:  STA     RCRC2   ; SAVE SECOND CRC CHARACTER
092C DB04               IN      MCR     ; RESET REQUEST TO SEND
092E E6FD               ANI     0FFH-RTS
0930 D304               OUT     MCR
0932 3A4310             LDA     INTFLAG         ; INDICATE RECEIVE ROUTINE
0935 E6FE               ANI     0FFH-RXBUSY     ; IS NOT ACTIVE
0937 324310             STA     INTFLAG
093A 214309             LXI     H,EXIT  ; IGNORE ALL RECEIVE INTERRUPTS
093D 224A10             SHLD    RNEXT
0940 C34309             JMP     EXIT
0943 C1         EXIT:   POP     B
0944 D1                 POP     D
0945 E1                 POP     H
0946 F1                 POP     PSW
0947 FB                 EI
0948 C9                 RET
                ***********************************************************
                ; TRANSMIT INTERRUPT ROUTINES

0949 2A4C10     TXINT:  LHLD    TNEXT   ; DISPATCH ADDRESS IN TNEXT
094C E9                 PCHL

094D 3E16       TSTART: MVI     A,SYN   ; OUTPUT A SYN CHARACTER
094F D300               OUT     THR
0951 215A09             LXI     H,TDLE1 ; NEXT SEND A DLE
0954 224C10             SHLD    TNEXT
0957 C34309             JMP     EXIT

095A 3E10       TDLE1:  MVI     A,DLE   ; OUTPUT A DLE
095C D300               OUT     THR
095E 216709             LXI     H,TSTX  ; NEXT OUTPUT START OF TEXT
0961 224C10             SHLD    TNEXT
0964 C34309             JMP     EXIT

0967 3E02       TSTX:   MVI     A,STX   ; OUTPUT START OF TEXT
```

```
0969 D300               OUT     THR
096B 217409             LXI     H,TDATA ; NEXT FUNCTION HANDLES TEXT DATA
096E 224C10             SHLD    TNEXT
0971 C34309             JMP     EXIT

0974 211D10     TDATA:  LXI     H,OUTCOUNT      ; MORE DATA IN BUFFER?
0977 7E                 MOV     A,M
0978 B7                 ORA     A
0979 CA9709             JZ      TDATA1  ; NO, BUFFER EMPTY
097C 35                 DCR     M
097D 2A1A10             LHLD    LBOP
                        INCLB   1
0980+3E01               MVI     A,1
0982+DF                 RST     3
0983 221A10             SHLD    LBOP    ; LBOP = LBOP+1
0986 7E                 MOV     A,M
0987 D300               OUT     THR     ; OUTPUT DATA AT LBOP
0989 FE10               CPI     DLE     ; IS IT SAME AS DLE?
098B C24309             JNZ     EXIT    ; NO
098E 21A409             LXI     H,TDLE  ; TRANSMIT ANOTHER DLE
0991 224C10             SHLD    TNEXT   ; TO MAKE TRANSPARENT
0994 C34309             JMP     EXIT
0997 3E10       TDATA1: MVI     A,DLE   ; OUTPUT A DATA LINK ESCAPE
0999 D300               OUT     THR
099B 21B109             LXI     H,TETX  ; NEXT SEND END OF TEXT
099E 224C10             SHLD    TNEXT
09A1 C34309             JMP     EXIT

09A4 3E10       TDLE:   MVI     A,DLE   ; SEND DATA LINK ESCAPE
09A6 D300               OUT     THR
09A8 217409             LXI     H,TDATA ; AND GO BACK TO TRANSPARENT MODE
09AB 224C10             SHLD    TNEXT
09AE C34309             JMP     EXIT

09B1 3E03       TETX:   MVI     A,ETX   ; SEND END OF TEXT
09B3 D300               OUT     THR
09B5 21BE09             LXI     H,T1CRC ; NEXT SEND FIRST CRC CHARACTER
09B8 224C10             SHLD    TNEXT
09BB C34309             JMP     EXIT

09BE 3A4910     T1CRC:  LDA     TCRC1   ; SEND FIRST CRC CHARACTER
09C1 D300               OUT     THR
09C3 21CC09             LXI     H,T2CRC ; NEXT SEND SECOND CRC CHARACTER
09C6 224C10             SHLD    TNEXT
09C9 C34309             JMP     EXIT

09CC 3A4810     T2CRC:  LDA     TCRC2   ; SEND SECOND CRC CHARACTER
09CF D300               OUT     THR
09D1 21DA09             LXI     H,TPAD  ; SEND TRAILING PAD CHARACTER
09D4 224C10             SHLD    TNEXT
09D7 C34309             JMP     EXIT

09DA 3EFF       TPAD:   MVI     A,PAD   ; SEND TRAILING PAD AFTER CRC
09DC D300               OUT     THR
09DE 3A4310             LDA     INTFLAG ; MARK TRANSMIT NOT BUSY
09E1 E6FD               ANI     0FFH-TXBUSY
09E3 324310             STA     INTFLAG
09E6 CD7608             CALL    DISABLETX
09E9 C34309             JMP     EXIT
                ***********************************************************
                ; CRC CALCULATION ROUTINE
                ; INCLUDES BYTE IN ACCUMULATOR IN CRC CALCULATION
09EC F5         CALCCRC:PUSH    PSW
09ED 0608               MVI     B,8
09EF 4F                 MOV     C,A
09F0 2A4410             LHLD    CRC
09F3 =          CALCCRC1:EQU    $
09F3 79                 MOV     A,C
09F4 07                 RLC
09F5 4F                 MOV     C,A
09F6 7D                 MOV     A,L
09F7 17                 RAL
```

```
09F8 6F                 MOV     L,A
09F9 7C                 MOV     A,H
09FA 17                 RAL
09FB 67                 MOV     H,A
09FC D2070A             JNC     CALCCRC2
09FF 7C                 MOV     A,H
0A00 EE80               XRI     80H
0A02 67                 MOV     H,A
0A03 7D                 MOV     A,L
0A04 EE05               XRI     05H
0A06 6F                 MOV     L,A
0A07 =        CALCCRC2:EQU      $
0A07 05                 DCR     B
0A08 C2F309             JNZ     CALCCRC1
0A0B 224410             SHLD    CRC
0A0E F1                 POP     PSW
0A0F C9                 RET
              **********************************************************
              ; RECEIVE SIDE DISPATCH ROUTINES

0A10 2A4E10   DISPRX: LHLD      RDISP   ; GO TO RECEIVE DISPATCH ROUTINE
0A13 E9                 PCHL

0A14 2A0610   WAITTB: LHLD      OTBE    ; TERMINAL BUFFER CUSHION FREE?
0A17 EB                 XCHG
0A18 2A0410             LHLD    CTBIE
                        COMPARE         ; COMPARE DE TO HL
0A1B+EF                 RST     5
0A1C CA280A             JZ      WAITTB1 ; SAME, BUFFER AVAILABLE
                        INCTB   CUSHION ; IS CUSHION FREE?
0A1F+3EFF               MVI     A,CUSHION
0A21+D7                 RST     2
0A22 DA120B             JC      DISPTX  ; TO TRANSMIT ROUTINE DISPATCHER
0A25 2A0410             LHLD    CTBIE   ; POINT TBIP JUST AHEAD OF DATA
              WAITTB1:INCTB     1
0A28+3E01               MVI     A,1
0A2A+D7                 RST     2
0A2B 220810             SHLD    TBIP
0A2E 3E00               MVI     A,0     ; ZERO COUNT FOR RECEIVE ROUTINE
0A30 321C10             STA     BUFCOUNT
0A33 F3                 DI
0A34 3A4310             LDA     INTFLAG ; RECEIVE ROUTINE IS ACTIVE
0A37 F601               ORI     RXBUSY
0A39 324310             STA     INTFLAG
0A3C FB                 EI
0A3D 219008             LXI     H,RSTART        ; START RECEIVING
0A40 224A10             SHLD    RNEXT
0A43 DB04               IN      MCR     ; SET RTS SO OTHER END WILL SEND
0A45 F602               ORI     RTS
0A47 D304               OUT     MCR
0A49 21500A             LXI     H,WAITRX        ; WAIT FOR BLOCK
0A4C 224E10             SHLD    RDISP
0A4F C9                 RET

0A50 3A4310   WAITRX: LDA       INTFLAG ; IS RECEIVER STILL BUSY?
0A53 E601               ANI     RXBUSY
0A55 C2120B             JNZ     DISPTX  ; YES, GO TO TRANSMIT DISPATCHER
0A58 3A5210             LDA     DFLAG   ; GET DISPATCHER FLAG
0A5B E601               ANI     CRCTX   ; IS CRC ROUTINE BUSY?
0A5D C2120B             JNZ     DISPTX  ; YES, GO TO TRANSMIT DISPATCHER
0A60 211C10             LXI     H,BUFCOUNT      ; COUNT OF BYTES RECEIVED
0A63 7E                 MOV     A,M
0A64 2A0410             LHLD    CTBIE   ; POINT TO CURRENT INPUT ENTRY
0A67 77                 MOV     M,A     ; PUT COUNT IN BUFFER HEADER
                        INCTB   1       ; POINT JUST BEFORE DATA AREA
0A68+3E01               MVI     A,1
0A6A+D7                 RST     2
0A6B 220810             SHLD    TBIP
0A6E 210000             LXI     H,0     ; INITIALIZE CRC VALUE
0A71 224410             SHLD    CRC
0A74 217B0A             LXI     H,RXCRC ; NEXT CALCULATE CRC
0A77 224E10             SHLD    RDISP
```

```
0A7A C9                 RET

0A7B 211C10   RXCRC:  LXI     H,BUFCOUNT      ; MORE DATA TO INCLUDE?
0A7E 7E               MOV     A,M
0A7F B7               ORA     A
0A80 CA970A           JZ      RXCRC1  ; NO, GO TO INCLUDE CONTROL CHARS
0A83 35               DCR     M       ; DECREMENT COUNT
0A84 2A0810           LHLD    TBIP    ; UPDATE POINTER TO NEXT POSITION
                      INCTB   1
0A87+3E01             MVI     A,1
0A89+D7               RST     2
0A8A 220810           SHLD    TBIP
0A8D 7E               MOV     A,M     ; GET DATA BYTE IN A
0A8E CDEC09           CALL    CALCCRC ; INCLUDE IT IN CRC CALCULATION
0A91 FE10             CPI     DLE     ; WAS IT DATA LIKE A DLE?
0A93 CCEC09           CZ      CALCCRC ; DO ANOTHER FOR TRANSPARENCY
0A96 C9               RET             ; RETURN TO LIP
0A97 3E10     RXCRC1: MVI     A,DLE   ; INCLUDE DLE AND ETX IN CRC
0A99 CDEC09           CALL    CALCCRC
0A9C 3E03             MVI     A,ETX
0A9E CDEC09           CALL    CALCCRC
0AA1 21A80A           LXI     H,CHECKCRC      ; NEXT CHECK THE CRC
0AA4 224E10           SHLD    RDISP   ; GO INCLUDE CRC CHARS RECEIVED
0AA7 C9               RET

0AA8 3A4710   CHECKCRC:LDA    RCRC1   ; INCLUDE RECEIVED CRC CHARACTERS
0AAB CDEC09           CALL    CALCCRC
0AAE 3A4610           LDA     RCRC2
0AB1 CDEC09           CALL    CALCCRC
0AB4 21BB0A           LXI     H,CHKFIN        ; NEXT CHECK IF CRC IS GOOD
0AB7 224E10           SHLD    RDISP
0ABA C9               RET

0ABB 2A4410   CHKFIN: LHLD    CRC     ; GET CALCULATED CRC
0ABE 7D               MOV     A,L     ; IS IT ZERO?
0ABF B4               ORA     H
0AC0 C2D00A           JNZ     CHKFIN1 ; NO, GO RESTART RECEIVE OPERATION
0AC3 DB04             IN      MCR     ; YES, GOOD CRC, FLIP DTR
0AC5 EE01             XRI     DTR
0AC7 D304             OUT     MCR
0AC9 21D70A           LXI     H,RPROC ; PROCESS CHECKED BLOCK
0ACC 224E10           SHLD    RDISP
0ACF C9               RET
0AD0 21140A   CHKFIN1:LXI     H,WAITTB        ; BAD CRC, TRY AGAIN
0AD3 224E10           SHLD    RDISP
0AD6 C9               RET

              ; THIS ROUTINE SHOULD PROCESS THE BUFFER PREFIX
              ; TEMPORARILY IT ONLY PASSES THE BUFFER TO THE LIP
              ; AND HANDLES CONNECT/DISCONNECT
0AD7 2A0410   RPROC:  LHLD    CTBIE   ; IS COUNT 7 OR MORE?
0ADA 7E               MOV     A,M
0ADB FE07             CPI     7
0ADD DA020B           JC      RPROC2  ; NO, PASS TO LIP
                      INCTB   8       ; POINT TO SEE IF CONNECT OR
0AE0+3E08             MVI     A,8
0AE2+D7               RST     2
0AE3 7E               MOV     A,M     ; DISCONNECT
0AE4 FE01             CPI     'A'-40H ; IS IT CONNECT?
0AE6 C2F30A           JNZ     RPROC1  ; NO, GO TO TEST FOR DISCONNECT
0AE9 3E00             MVI     A,0     ; 0 FOR CONNECT
0AEB F7               RST     6       ; COMMUNICATE REQUEST TO LIP
0AEC 21140A           LXI     H,WAITTB        ; DON'T PASS THIS ENTRY
0AEF 224E10           SHLD    RDISP
0AF2 C9               RET
0AF3 FE02     RPROC1: CPI     'B'-40H ; IS IT DISCONNECT?
0AF5 C2020B           JNZ     RPROC2  ; NO, PASS TO LIP
0AF8 3E01             MVI     A,1     ; YES, 1 FOR DISCONNECT
0AFA F7               RST     6       ; COMMUNICATE REQUEST TO LIP
0AFB 21140A           LXI     H,WAITTB        ; DON'T PASS THIS ENTRY
0AFE 224E10           SHLD    RDISP
0B01 C9               RET
```

```
0B02 2A0810     RPROC2: LHLD    TBIP    ; UPDATE CURRENT INPUT ENTRY
                        INCTB   1
0B05+3E01               MVI     A,1
0B07+D7                 RST     2
0B08 220410             SHLD    CTBIE
0B0B 21140A             LXI     H,WAITTB        ; NOW GO GET ANOTHER ONE
0B0E 224E10             SHLD    RDISP
0B11 C9                 RET
                ************************************************************
                ; TRANSMIT SIDE DISPATCH ROUTINES

0B12 2A5010     DISPTX: LHLD    TDISP   ; GO TO TRANSMIT DISPATCH ROUTINE
0B15 E9                 PCHL

                ; THIS ROUTINE SHOULD PROCESS THE BUFFER PREFIX BUT
                ; TEMPORARILY IT ONLY PASSES THE BUFFER TO THE HOST
                WAITLIP:NEXTIN  ; IS THERE A BUFFER ENTRY FROM THE LIP?
0B16+E7                 RST     4
0B17 C8                 RZ              ; NO, RETURN
0B18 3A5210             LDA     DFLAG   ; INDICATE TX SIDE USING CRC
0B1B F601               ORI     CRCTX   ; ROUTINES
0B1D 325210             STA     DFLAG
0B20 7E                 MOV     A,M     ; GET DATA LENGTH FROM HEADER
0B21 321D10             STA     OUTCOUNT ; FOR CRC CALCULATION ROUTINE
                        INCLB   3       ; POINT JUST BEFORE DATA AREA
0B24+3E03               MVI     A,3
0B26+DF                 RST     3
0B27 221A10             SHLD    LBOP    ; FOR CRC CALCULATION ROUTINE
0B2A 210000             LXI     H,0     ; INITIALIZE CRC VALUE
0B2D 224410             SHLD    CRC
0B30 21370B             LXI     H,TXCRC ; NEXT START CRC CALCULATION
0B33 225010             SHLD    TDISP
0B36 C9                 RET

0B37 211D10     TXCRC:  LXI     H,OUTCOUNT    ; ANY MORE DATA TO INCLUDE?
0B3A 7E                 MOV     A,M
0B3B B7                 ORA     A
0B3C CA530B             JZ      TXCRC1  ; NO, GO TO INCLUDE CONTROL CHARS
0B3F 35                 DCR     M       ; DECREMENT COUNT
0B40 2A1A10             LHLD    LBOP    ; UPDATE POINTER TO NEXT POSITION
                        INCLB   1
0B43+3E01               MVI     A,1
0B45+DF                 RST     3
0B46 221A10             SHLD    LBOP
0B49 7E                 MOV     A,M     ; GET DATA BYTE IN A
0B4A CDEC09             CALL    CALCCRC ; INCLUDE IT IN CRC CALCULATION
0B4D FE10               CPI     DLE     ; WAS IT DATA LIKE A DLE?
0B4F CCEC09             CZ      CALCCRC ; DO ANOTHER FOR TRANSPARENCY
0B52 C9                 RET             ; RETURN TO LIP
0B53 3E10       TXCRC1: MVI     A,DLE   ; INCLUDE DLE AND ETX IN CRC
0B55 CDEC09             CALL    CALCCRC
0B58 3E03               MVI     A,ETX
0B5A CDEC09             CALL    CALCCRC
0B5D 21640B             LXI     H,CRCFIN ; NEXT TO FINISH CRC FOR SENDING
0B60 225010             SHLD    TDISP
0B63 C9                 RET

0B64 3E00       CRCFIN: MVI     A,0     ; FINISH OFF CRC CALCULATION FOR
0B66 CDEC09             CALL    CALCCRC ; TRANSMISSION
0B69 CDEC09             CALL    CALCCRC
0B6C 2A4410             LHLD    CRC     ; SAVE CALCULATION FOR TRANSMIT
0B6F 224810             SHLD    TCRC2
0B72 21790B             LXI     H,STARTTX       ; NEXT, START TRANSMITTING
0B75 225010             SHLD    TDISP           ; THE BLOCK
0B78 C9                 RET

0B79 3A4210     STARTTX:LDA     MSRSAVE ; SAVE CURRENT DSR LEVEL
0B7C E620               ANI     DSR
0B7E 67                 MOV     H,A
0B7F 3A5210             LDA     DFLAG
0B82 E6DE               ANI     0FFH-CRCTX-DSR  ; INDICATE CRC ROUTINE
0B84 B4                 ORA     H               ; NOT IN USE AND SAVE
```

```
0B85 325210             STA     DFLAG           ; DSR IN DFLAG
0B88 214D09             LXI     H,TSTART        ; SET UP INITIAL XMIT
0B8B 224C10             SHLD    TNEXT           ; INTERRUPT ROUTINE
0B8E 2A1610             LHLD    OLBE    ; POINT TO DATA TO TRANSMIT
0B91 7E                 MOV     A,M     ; GET COUNT
0B92 321D10             STA     OUTCOUNT
                        INCLB   3
0B95+3E03               MVI     A,3
0B97+DF                 RST     3
0B98 221A10             SHLD    LBOP
0B9B F3                 DI              ; DISABLE INTERRUPTS
0B9C 3A4310             LDA     INTFLAG ; INDICATE TRANSMIT BUSY
0B9F F602               ORI     TXBUSY
0BA1 324310             STA     INTFLAG
0BA4 3A4210             LDA     MSRSAVE ; IS CTS UP?
0BA7 E610               ANI     CTS
0BA9 CAB20B             JZ      STARTTX1 ; DON'T ENABLE TRANSMIT INTRPT
0BAC DB01               IN      IER     ; YES, ENABLE TRANSMIT INTERRUPTS
0BAE F602               ORI     ETBEI
0BB0 D301               OUT     IER
0BB2 FB         STARTTX1:EI             ; ENABLE INTERRUPTS
0BB3 21BA0B             LXI     H,WAITTX ; WAIT FOR TRANSMIT TO FINISH
0BB6 225010             SHLD    TDISP
0BB9 C9                 RET

0BBA 3A4310     WAITTX: LDA     INTFLAG ; TRANSMITTER INTERRUPTS ENABLED?
0BBD E602               ANI     TXBUSY
0BBF C0                 RNZ             ; YES, RETURN
0BC0 211027             LXI     H,ACKTO ; NO, SET UP FOR TIMEOUT
0BC3 224010             SHLD    WAIT    ; INITIALIZE ACKNOWLEDGE TIMEOUT
0BC6 21CD0B             LXI     H,WAITACK       ; NEXT WAIT FOR ACKNOWLEDGE
0BC9 225010             SHLD    TDISP
0BCC C9                 RET

                WAITACK:
0BCD 215210             LXI     H,DFLAG ; IS DSR SAME AS BEFORE?
0BD0 3A4210             LDA     MSRSAVE
0BD3 AE                 XRA     M
0BD4 E620               ANI     DSR
0BD6 C2EA0B             JNZ     WAITACK1        ; NO, BLOCK ACKNOWLEDGED
0BD9 2A4010             LHLD    WAIT    ; YES, DECREMENT TIMEOUT COUNT
0BDC 2B                 DCX     H
0BDD 224010             SHLD    WAIT
0BE0 7C                 MOV     A,H     ; IS TIME OVER?
0BE1 B5                 ORA     L
0BE2 C0                 RNZ             ; NO, RETURN
0BE3 21790B             LXI     H,STARTTX ; YES, TIMED OUT, SO SEND AGAIN
0BE6 225010             SHLD    TDISP
0BE9 C9                 RET
                WAITACK1:
0BEA 21160B             LXI     H,WAITLIP  ; GOOD ACK, GET ANOTHER BUFFER
0BED 225010             SHLD    TDISP      ; FROM LIP
0BF0 C9                 RET

0BF1                    END
```

LINK LEVEL ADDRESS MECHANISMS IN AMATEUR RADIO PROTOCOLS

by

H. S. Magnuski, KA6M
311 Stanford Avenue
Menlo Park, CA 94025

In October, 1982, agreement was reached on a new Link Level protocol for amateur packet radio networks. One of the unique aspects of the protocol is the set of address fields used at the beginning of each frame. This paper reviews the types of addressing used prior to the adoption of the new standard, and explains in detail how the new address mechanism works.

On October 9th, 1982, representatives of the active packet radio user groups throughout the U.S. agreed to establish a new standard for Link Level connections in the packet radio service. This agreement unified the development activities of several diverse groups, and provides for a point-to-point protocol which can be used both in terrestrial and satellite networking. This paper will review how addressing was done prior to the adoption of the new standard, and will then describe, in some detail, how the new address mechanism works, and what advantages and disadvantages are gained by adopting it.

1.0 Background of HDLC Numerical Addressing

The first byte of an HDLC frame following the opening flag is always an address byte. This byte permits a maximum of 256 stations in a network to be addressed, and sometimes less, as byte 00 hex is often reserved for special purposes, and byte FF hex is usually a broadcast (all-stations-addressed) address. Since 254 stations is a reasonable number for active stations on a single frequency metropolitan area net (metronet), the initial implementor of HDLC-oriented transmission, Doug Lockhart, VE7APU, decided that no more complicated addressing scheme was required and he programmed his Terminal Node Controller board to deal with a single byte address. Doug's scheme initially utilized dynamic address assignment, and works as follows:

When a station initially came on the air, it would send a special sign-on packet to a central control station, which would assign the next available numerical address to the new user. If the user logged off or timed out, the address would be put back into the free pool and handed to the next station to sign in. Once assigned an address, the station would use that address in all outgoing packets, and the central control station would know by that address who originated each packet in the metronet. All traffic flowed through the central control station, so there was no need for one remote to address another.

Several groups in Canada and the U.S. desired to use Doug's code without a central station, and so the following form of static addressing was adopted:

Each user in a specific geographical area was assigned a numeric address, and that address was hard-wired into the outgoing address field of each transmitted frame. After a connection was made, the receiving station would lock onto the transmitting station's address, and all other packets on the channel would be discarded.

The author implemented a variation of this idea that would allow a simplex repeater to be used as part of a metronet. Some range of the address space, say 80 to 9F hex would be dedicated as repeater input addresses, and in repeating the packet the repeater would transform the address to the range A0 to BF. Thus a station would use address 01 for point-to-point work, and would use addresses 81 and A1 for repeater uplink and downlink, respectively.

The dynamic and static addressing schemes described above served the early packet experimenters well, but it soon became clear that there were significant problems with each of the methods.

The dynamic addressing scheme failed if the central control station failed, and many users did not want to be dependent on a single central repeater. The scheme did not allow for multiple repeaters serving a given area, and failed if there was any overlap in central or remote station RF domains.

The static assignment method was just as bad, requiring an individual or club to prescribe addresses, and limiting the number of users to 62 if a repeater was involved. Visitors to an area might conflict with already assigned addresses, and using statically assigned addresses on a global satellite channel would be impossible.

Since neither of these methods would be workable over the long term, several proposals were put forward to correct the problem.

2.0 HDLC Extended Addressing

It became clear that if one could substitute a callsign for the hardwired numeric code, some of the addressing problems mentioned above would go away. The radio callsign is a unique identifier, and could be used to tag each packet being transmitted. So Terry Fox and others at AMRAD in Washington, D.C., proposed to utilize the extended addressing feature in HDLC to incorporate the transmitting callsign.

The HDLC standard permits an arbitrary extension of the address field through utilization of the least significant bit of the address byte as an extension flag. Basically, if the least significant bit (lsb) is zero, then the next byte forms part of the address field too, and one continues to extend the address field until a byte is found where the lsb is set to one.

This idea would be fairly easy to implement, and did not require much overhead in each packet. It also did not violate any standards related to the HDLC spec. Unfortunately, there were deficiencies in the method, particularly with regard to point-to-point addressing in a network, and flexible use of multiple repeaters in one RF domain. With just one address, a station could not target the receiver of an outgoing packet. Also, no provisions were made for dealing with multiple repeaters on the same frequency, and it became clear that this approach would have to be revised.

3.0 The New Link Level Standard

When the packet radio groups got together in October there was clearly an urgent need to agree upon a uniform link level standard, and to adopt one which would be sufficiently flexible to accomodate a variety needs and environments.

Two proposals were put forward at the meeting, and it turned out that there was more in common in the proposed solutions than there were differences. The first proposal was by the AMRAD and New Jersey packeteers, and it was a result of extensive discussions on how to implement an X.25 type of service suitable for the radio community. This proposal, called AX.25, called for a destination and source callsign in an extended address field. The other proposal was from the author, and it also specified use of two or optionally three callsigns in the packet, but positioned them immediately after the control field. Each party was willing to compromise, the callsigns were kept in the extended address field, an optional repeater address field was added to the AX.25 spec, definitions for bits in the sub-station and protocol ID fields were rearranged, and thus emerged the new standard protocol.

4.0 Components of the Address Field

As stated before, the standard protocol relies on the HDLC extended addressing bit to mark the end of the address field. The address field is composed of two or three seven-byte callsign address fields constructed in the following manner:

```
CHAR1 CHAR2 CHAR3 CHAR4 CHAR5 CHAR6 SSID
```

where each call is left justified in the field, padded with blanks, uppercase, and each CHAR is shifted left so that the lsb is zero, except for the last byte of the address fields, where the lsb is set to one, indicating the end end of HDLC extended addressing. The first callsign field is for the destination or receiving station, the second callsign field is for the source or transmitting station, and the optional third field is for a repeater.

The SSID byte is of the form:

```
R 11 SSSS X
```

where R is the "repeated" bit, set to one only in the optional repeater address field when the packet has been repeated. The '11' field is reserved and set to ones. It may be used for control purposes if required by a local area net. The SSSS field is the sub-station code, normally all zeros except for situations where a person has more than one station on the air. The X is normally zero, unless this field is the last of the callsigns, in which case it is set to one, indicating the end of the variable length address field.

5.0 Uses of the Address Fields

The simplest case is a point-to-point connection between two stations. Station A puts CALL B in the destination field, and CALL A in the source field. Similarly, Station B puts CALL A in the destination field and CALL B in the source field. When A receives a packet it first verifies that the frame check sequence (FCS) is correct. Next, it scans the address field to determine if the address field length is either 14 or 21 bytes. Assuming that the field is exactly 14 bytes long, it checks for CALL A in the destination field and for CALL B in the source field, and discards the packet if there is not a correct match.

If a repeater is being used, the optional third address field will be utilized, and the following sequence of actions will take place:

The transmitting station extends the address field to 21 bytes and places the callsign of the desired repeater into the third field. The I've-been-repeated bit (R bit) must be set to zero. The repeater is constantly monitoring the channel for packets and discards all packets which do not have a correct FCS or do not have exactly 21 bytes in the address field. If the third address field is present and if the call exactly matches the repeater's call and SSID, and if the R bit is zero, then the repeater sets the R bit to one and retransmits the packet. The receiving station determines that the address field is exactly 21 bytes and checks that the R bit is set to one. If the R bit is zero it has received an uplink packet and should discard it. The receiving station then checks for proper source and destination callsigns, and accepts the packet on a proper match.

The receiving station knows how to reply to an incoming transmission because an examination of the address field will inform it whether or not a repeater was used, and if one was used, its callsign. This information is used to construct the reverse path to the transmitting station.

6.0 Limitations

This protocol is intended only for point-to-point connections directly or through a single repeater. The situation where multiple repeats are required is not covered and is the task of higher level protocols.

The protocol also does not specify when the reverse path should be constructed. An implementor may decide to establish the reverse path only on information contained in the initial connect (SABM) packet. If so, and if the connector changes paths during a session, the two stations might be using different forward and reverse paths to talk to each other.

The call signs add overhead to each packet (20 bytes at most), and these extra bytes will obviously slow down throughput. In doing link calculations, however, one finds that radio and modem turnaround delays are probably the most significant factors affecting overall efficiency, and that the extra overhead of the callsigns are justified in terms of the elimination of the requirement for a central control station, or elimination of any kind of connection-state information in the repeater.

7.0 Summary

The address mechanisms in this new protocol are fairly simple to implement and provide for a point-to-point connection in local area metronets, in backbone terrestrial networks, and in satellite channels. The addition of a single repeater option makes the protocol useful for communications in a limited geographical area. The protocol will serve as a building block for higher level connection-oriented protocols such as AX.25, and can be easily used for connectionless, datagram-oriented protocols such as IP/TCP.

REAL-TIME LOW LEVEL SOFTWARE ON THE TAPR TNC

Margaret Morrison, KV7D
4301 E. Holmes Street
Tucson, Arizona 85711
602-325-4775

Introduction

This paper describes the low-level assembly language routines (LLR) of software released with the Tucson Amateur Packet Radio (TAPR) Beta Test terminal node controllers (TNCs). The primary functions performed by these routines are initialization of peripheral devices and data in RAM, maintaining input and output (I/O) buffers, servicing interrupts from peripheral devices, handling nonvolatile RAM data storage and retrieval, and calibration and checkout routines. Entry points are provided which are appropriate to the subroutine calling sequence of the Pascal compiler used for the high-level routines (HLR). In addition, a low-level debug program provides capability for direct access to peripherals, inspection of RAM and ROM locations, and execution of temporary code in RAM. The present LLR code occupies about six kilobytes of ROM.

Initialization

On receipt of a RESTART interrupt by the processor, control passes to the low-level initialization routine. This section first sets up the hardware stack, and then reads the on-board DIP switches which direct the initialization of user-settable parameters. These parameters, which can be set to their default values stored in ROM, or read from the 64 x 4 nonvolatile RAM, are decoded and expanded. The program initializes buffer pointers, counters, timers, and status flags, as well as non-permanent user-settable parameters. The peripheral chips are initialized and checked to verify that they can be commanded. If the UART can not be commanded, the program is aborted and another reset is attempted. Failure of other peripherals results in diagnostic messages. The VIA timer functions are configured so that Timer 1 acts as a pulse generator for the HDLC controller and Timer 2 generates interrupts for software timing functions. The initialization section terminates by enabling interrupts and typing a sign-on message, and passes control to the HLR.

Buffer Management

One of the primary tasks of the LLR is maintaining the I/O buffers. At any time there are four active I/O buffers, input and output buffers for terminal and radio data. An echo buffer, which is configured identically with the terminal output buffer may also be present. Data received from peripherals (terminal and radio interface) are placed into input buffers which are read upon calls from HLR. Data placed into output buffers by calls from HLR are passed to the peripherals under interrupt control. Each buffer has an insertion pointer, which is updated as data are added to the buffer so that it points to the next available cell, and a removal pointer which points to the next cell to be read. All buffers are "circular", meaning that each time a pointer is advanced it is compared with the top of the buffer space, and moved if necessary to the bottom of the buffer space. A buffer is "empty" when the insertion and removal pointers are the same, and "full" when the insertion pointer points to the next cell below the removal pointer. The incoming and outgoing packet buffers contain, as the first bytes of each packet, the byte count of the packet.

In addition to the two basic pointers, input buffers have other markers to facilitate input editing, and output buffers have space-available counters for use by routines writing to these buffers.

A pointer to the beginning of an incoming packet serves two purposes. First, it allows an incoming packet to be purged in case of a reception error (invalid frame-check sequence or insufficient buffer space). Second, it facilitates storing the byte count of the packet upon successful reception.

The terminal input buffer management is rather complex, and operates in three different modes. In "command mode," character and line editing functions are in effect, and a pointer to the beginning of the current line insures that deletion past this point will not take place in case of rapid terminal input or slow input processing. In "conversation mode," an additional pointer marks the beginning of the current packet (actually the data portion of the packet), and an additional editing feature allows the user to cancel the current packet. Packets remain in the input buffer until they have been acknowledged, at which time the buffer is updated by a call from HLR. A completed packet is signalled by the receipt of a packet-terminating character, and this character is placed in the buffer as a marker for the end of the packet. In order to prevent commands from interfering with partially typed packets, separate input buffers are maintained, and the pointers for the active buffers are swapped when the mode is changed. This has the effect of moving time-consuming decisions from the interrupt dependent program to subroutines called from HLR.

The third input mode is "transparent mode," in which all characters received from the terminal are transmitted. Packets are terminated on the basis of number of characters or occurrence of a timeout. Characters between the removal pointer and the packet marker are divided into maximum-length packets, with the remainder going into a short packet. In order to avoid ambiguity as to the composition of a packet, the packet marker is not updated until all outstanding packets have been acknowledged and cleared from the input buffer.

Interrupt Service

Although the 6809 microprocessor supports two levels of hardware interrupt, a fast interrupt (FIRQ) and a normal interrupt (IRQ), only the IRQ is implemented on the TAPR TNC. All peripheral IRQ lines are wire-ORed together into the IRQ input to the processor. Interrupt outputs from each peripheral can be disabled independently without affecting the processor's response to IRQ. Upon receipt of IRQ, the processor transfers execution to a routine which examines each device in turn and passes control to a routine specified in a dispatch table. For maximum flexibility, the interrupt dispatch table is stored in RAM. The addresses in the table are initialized during the startup procedure, and are changed as the TNC operates in different modes.

The peripheral devices are assigned priorities according to the order in which the dispatch routine checks their status. The priority of service is

1. UART (terminal) input
2. UART output
3. Timer interrupt
4. All HDLC (radio interface) interrupts

Interrupts from the parallel I/O port are not enabled in the initial software release; when they they will be assigned lowest priority. The HDLC is currently assigned lowest priority because the interrupt service is quite complex and the chip will generate spurious interrupts at a high rate under noisy conditions. The UART was placed ahead of the timer, since the 6522 timer mode used provides a mechanism for compensating for delayed interrupt servicing. The UART output ought properly to be assigned a lower priority, but it was placed after the input service for convenience, since input and output status are given in the same register. Parallel port I/O must be assigned the lowest priority to insure that it does not interfere with other interrupt service.

As a diagnostic tool, the interrupt dispatch routine writes signals reflecting interrupt conditions to the parallel port, which is configured as 16 output bits for the initial software release.

Terminal I/O

The terminal input interrupt service normally consists of two parts: an initial routine during which interrupts remain disabled, and a subsequent unprotected routine. The protected routine reads a character from the UART data register and places it in a temporary holding buffer, after which interrupts are enabled. This is because the input editing and echoing function is enabled upon receipt of each character, and this can be a complex procedure in some cases. Depending on the input mode, the character may be tested against an assortment of special characters, and this may result in flow control action or change of input mode. Characters which terminate packets or command lines require that pointers be updated and flags set for other routines. Input editing characters cause immediate update of the input buffer. Finally, an echo routine is called, which places characters in an echo output buffer. Input characters are not echoed directly, since adequate terminal support sometimes requires that more than one character be echoed for each character input. In particular, automatic line feed after carriage return is a common requirement, and some hard-copy devices require many null characters following a carriage return. If the input buffer becomes nearly full, a request may be made to the output routine to transmit an XOFF character. Alternatively, a routine to produce the appropriate hardware flow control signals to the RS-232 interface may be called.

Special input service routines are used when the program operates in "transparent mode." Since there are no special characters, the input character is simply placed in the buffer. A timer may be started, and a flag will be set if a maximum packet count has been reached.

The UART output interrupt is disabled whenever there is no data to be sent, and any routine which requests output enables the interrupt. Any special treatment of characters, such as conversion to upper case or adding line feeds or nulls, is done by the routine which fills the output buffer, which also maintains a screen-width counter and inserts extra carriage returns as necessary. The output service routine checks for tasks to be performed in the following priority and performs the first task found.

1. Request to transmit XOFF character
2. Request to transmit XON character
3. Transmit characters in echo buffer
4. Transmit characters in output buffer

If all tasks are exhausted, the routine disables the output interrupt. This is not necessary, but as long as this interrupt is enabled, the UART generates an interrupt at regular intervals.

Timer Interrupts

The basic function of the timer interrupt is to update the software clocks. The interrupt is set to occur at 10 ms intervals, which provides good resolution for timing associated with the radio interface. For longer times, a "slow" clock is updated at one-second intervals. An additional 10 ms clock is used as a pseudo-random number generator for determining packet retry times.

In addition to the basic clock function, several tasks are performed under timer interrupt control. If a CW ID is in progress, the Morse Code routine is invoked every 60 ms to toggle the tone on or off as necessary. Otherwise, a variety of tasks to be performed after time lapses are checked. Packet transmissions are begun following an appropriate interval following detection of a carrier drop, and the ID is sent at regular intervals. If a CW ID is commanded manually, it is begun in this routine. A special set of routines is entered when the program runs in "transparent mode," and the appropriate routine is selected by reference to a status table. These routines mark packets for sending upon timeout of clocks which are started on character input, and set up guard times for the escape sequence for exiting this mode.

HDLC Interrupts

The WD-1933 HDLC controller generates interrupts for seven different conditions. Since any combination of interrupt conditions can be present, and most conditions are cleared by reading the status of the chip, all possibilities must be considered at every interrupt. The interrupt conditions are requests for service of input or output registers (DRQI or DRQO), transmission or receipt of end of message, with or without errors (XEOM or REOM), and change of state of carrier detect (DSC).

Transmit interrupts, DRQO, XEOM-ok, and XEOM-err, are handled according to a state table which indicates the progress of the packet in transmission. The appropriate action, as determined by the state table, is taken if any of these interrupts is detected, without reference to which condition was indicated. The only transmit error possible is under-run, or failure to service a DRQO. This condition should never occur in normal operation. Following complete transmission of a packet and while final flags are being sent, the routine checks for further packets to be transmitted before the transmitter is unkeyed. The CW ID may also be started from this routine.

Receive interrupts, DRQI, REOM-ok, and REOM-err, are handled according to the condition indicated by the chip status. In the event that more than one condition is present, a DRQI is assumed to precede an REOM, and the input register is read before closing the packet. If both REOM-ok and REOM-err are present, the error condition is disregarded. Causes of the error condition are aborted frame, invalid frame-check sequence, and over-run (failure to service DRQI). The cause of the error is not investigated and any partially received frame is cancelled. Another possible source of error is insufficient room in the input buffer for the incoming packet. If the input buffer becomes full, a flag is set and the routine continues to read incoming characters as they appear, but an REOM-ok is treated as if it were an REOM-err.

The DSC interrupt does not affect either transmit or receive operation. The carrier detect input of the HDLC controller is the demodulator lock-detect and is used to recognize a busy channel. The service routine maintains software flags which reflect the carrier detect state, and if the carrier is found to have dropped, a timer is started which indicates when the next transmission may be started.

Nonvolatile RAM Interface

A number of user-settable parameters are stored in a semi-permanent state in the Xicor NOVRAM. To make maximum use of the 32 bytes of storage, the data is encoded in a compact form. In order to be useful to the program, it must be translated. The information stored includes terminal attributes such as baud rate and parity; radio-link attributes such as baud rate, packet length, and transmitter keyup time; display features such as case conversion, echo mode, auto line feed, screen width, and nulls required. Special command characters for flow control, exit to command mode, and editing features are also stored, along with the station call sign. These parameters are changed by commands to HLR, which maintains the compressed copy of the parameters by calling LLR whenever a parameter is changed. This copy is overlaid on the nonvolatile storage as volatile RAM data, and becomes permanent when the "STORE" line is toggled. The nonvolatile RAM is controlled through the parallel I/O port of the 6522 VIA.

Calibration Routines

The hardware design of the TAPR TNC provides for on-board calibration routines. Jumpers are connected which allow the VIA Timer 2 to count the modulator or demodulator frequency to be calibrated. The VIA timers are reconfigured so that Timer 1 acts as a free-running count-down timer, counting at the 921.6 kHz system clock rate, and Timer 2 generates an interrupt after two cycles of the tone frequency being calibrated. By starting the timers simultaneously, Timer 1 can be used to count clock pulses for the duration of two periods of the frequency being calibrated. Two of the LED indicators are controlled in parallel with the microphone audio and HDLC reset lines, and are used in this routine as visual indicators of frequency deviation from the desired values. The timer routines in the interrupt dispatch table are replaced with special calibration service routines.

High-level Interface

Most of the entry points provided for HLR are concerned with buffer management. The I/O functions performed through these routines are: read a string from terminal or packet input buffer to a specified location; write a string from a specified location to terminal or packet output buffer; return character count for an outgoing packet in the terminal input buffer; update terminal input buffer; and return space-available count for output buffers. Other functions provided are: update nonvolatile RAM, temporary or permanent mode; enter calibration routine; force CW ID; and perform a soft reset. A special routine is called by HLR to notify LLR of a change of mode from "command mode" to "converse mode" or "transparent mode." All routines are called with the addresses of the arguments in processor registers D, X, Y, and U as needed.

Low-level Debug Program

A debugging facility is provided in the LLR, primarily as a software development tool. This program is invoked by a special user-settable character. In order to preserve as much information as possible about the system prior to entering the debugger, the active terminal input and output buffers are "frozen" upon entry by swapping the buffer pointers with a set of pointers used exclusively in the debugging program. All terminal input thenceforth is interpreted as commands to the debugger. These commands permit examination or modification of any addressable location, modification of the processor registers as they were upon entry to the program, or transfer of execution to any location. This allows test routines to be stored in RAM and executed. The user also has direct access to I/O addresses, and can observe the contents of I/O buffers without interfering with them.

DESIGNING THE TAPR TNC AUDIO INPUT FILTER

Margaret Morrison, KV7D and Dan Morrison, KV7B
4301 E. Holmes Street
Tucson, Arizona 85711
602-327-4775

Standard modulation for present terminal node controllers is Bell 202 compatible 1200 Hz / 2200 Hz phase-continuous FSK. Fig. 1 shows the typical spectral characteristics of such data. Notice that frequencies ranging from about 500 Hz to 2900 Hz are present in random NRZI data. One might guess that frequencies outside the central region from 1000 Hz to 2400 Hz could be eliminated with no degradation of demodulator performance. This turns out to be incorrect: the demodulator PLL needs this information to ensure timely response to data transitions. Thus, the ideal audio response over the link should be flat from below 500 Hz to over 2900 Hz.

In fact, the audio response of a typical FM 2-meter link looks like Fig. 2. This response curve shows dramatic rolloff over the frequency range of interest. Most of the rolloff is due to receiver audio characteristics and, in fact, some transmitted signals actually seem more like phase-modulation rather than FM, with high frequency emphasis. In many cases, the demodulator on the TAPR TNC cannot handle this rolloff, and successful demodulation of these signals requires reduced baud rates. We decided to put in a filter ahead of the demodulator to alleviate this problem.

The filter needed to take up as little space as possible, to be simple to modify, and most importantly, to be effective. In view of the fact that a variety of clock signals were already available on the TAPR TNC, we opted for a switched-capacitor filter design, since ease of modification and parts-count make this a clear winner in the design competition. We quickly realized that in order to produce frequency compensation over a very wide range at least two sections would be required. Lyle Johnson, our hardware guru, thus selected the National Semiconductor MF-10 dual section filter as the part of choice, leaving the design details up to us.

Filter Configuration

This marvelous part may be configured in a number of ways as first or second order filter sections depending on external components, and each section may be configured independently of the other. After some experimentation we chose two second order sections -- one set up as a highpass filter, the other as a lowpass filter. This choice optimizes both amplitude and phase response over the wide range of frequencies required.

The manufacturer's literature lists nine different recommended configurations, each yielding a variety of output options. The choice is quickly narrowed to one, mode 3, which provides both highpass and lowpass outputs, and which provides maximum flexibility in selecting filter parameters. In particular, the frequency parameter, f0, is adjustable via resistor choices rather than by clock frequency alone. This, as well as dynamic range considerations, directed us to select mode 3 for both sections. In addition, we chose the option which scales f0 to fclk/100 rather than to fclk/50 in order to have as high a clock rate as possible. This allows the switching noise, which occurs at the clock frequency, to be most easily removed by post-filtering.

Filter Response

The input section, configured as a highpass filter, is shown in Fig. 3a; the second section is configured as a lowpass filter, shown in Fig. 3b. The frequency response for these two sections may be expressed as functions of frequency, f, and the parameters f0, Q, and either highpass gain parameter GLP or lowpass gain parameter GHP. Let $\omega = jf/f0$, where $j^2 = -1$. Then, the highpass filter's (complex) frequency response is

$$h(f) = GHP \frac{\omega^2}{\omega^2 + \omega/Q + 1} \quad (1)$$

and the lowpass response is

$$h(f) = GLP \frac{1}{\omega^2 + \omega/Q + 1} \quad (2)$$

For either, the amplitude response may be determined as $|h|$, and the phase response as

$$\phi_{out} - \phi_{in} = \arctan\left[Im(h)/Re(h)\right] \quad (3)$$

The parameters f0H, QH, and f0L, QL are related to the clock frequency, and the programming resistors, R2, R3, and R4, by the following equations:

$$f0 = \frac{fclk}{100}\sqrt{\frac{R2}{R4}} \quad (4)$$

$$Q = \frac{R3}{R2}\sqrt{\frac{R2}{R4}} \quad (5)$$

where F0 is f0H or f0L and Q is QH or QL, depending on the section.

For the highpass section, the gain is GHP = -R2/R1, while for the lowpass section the gain is GLP = -R4/R1. Notice that for either section the input resistor, R1, only affects the overall gain and does not influence the shape of the response.

Optimization Procedure

The design task begins by choosing f0H, QH, f0L, and QL so as to produce as flat an overall filtered response as possible given the unfiltered response. These four parameters are determined by successive interactive computer optimization of the filtered response. A systematic search is made for the optimum parameters for each section alternately.

The optimization of either section consists of a nested iteration which determines its Q in an inner loop and its f0 in an outer loop. The other section's parameters remain fixed during this iteration. The inner loop calculates a least-squares straight line fit to the filtered audio amplitude response (expressed in dB), over a specified range of frequencies, and varies Q to produce a flat response, that is, best-fit slope of zero.

This procedure is written as a function whose value is the RMS deviation of the filtered response from the fit, and which also returns to the outer loop as arguments the value of Q, the slope, and the intercept of the fit. The outer procedure consists of a non-linear optimization (minimization) of the RMS error between the filtered response and the best-fitting frequency-indendent response, as a function of the f0 parameter.

The specified frequency range for the calculation can be varied to achieve best results. Typically, optimizing over the range 1000 Hz to 2400 Hz produces very good results, as rolloff begins well outside the range of optimization. It is helpful to be able to graphically display intermediate response curves during the optimization and to allow only inner-loop iterations on occasion.

Programming the Filter

The remaining work is to determine the eight resistor values (four in each section) which best implement the optimized response functions. Several considerations allow a unique choice to be made. First, for best dynamic range in mode 3, fclk/100 should be as near to f0 as possible. Since both sections run off the same clock, the optimum compromise clock frequency is the geometric mean,

$$\frac{fclk}{100} = \sqrt{f0H \cdot f0L} \quad (6)$$

In practice, on the TAPR TNC this meant picking the nearest available frequency to this (fclk = 115.2 kHz), since the main crystal clock frequency is a power of two times any of the available values. Having selected the proper clock frequency we are able to determine the resistance values. This is done in the following steps:

1. Notice that in either section the input resistor, R1, does not determine response shape -- only overall gain of the stage. Using Eqs. (4) and (5) determine the ratios R2 : R3 : R4 which affect the shape of the response.

2. Temporarily design each section so as to have unity amplitude response at 1200 Hz. Having thus determined the overall gain parameters GLP and GHP, determine the ratio of R1 to R2 for the highpass section and the ratio of R1 to R4 for the lowpass section based on the gain-resistor relationships.

3. Arbitrarily set the lowest resistance value in each section to 10 kilohms. This allows at least two of the eight resistors to be of garden variety. In addition, the resistances will be as low as possible while staying well above the minimum value recommended by the manufacturer (5 kilohms). Finish the design by using the ratios R1 : R2 : R3 : R4 to determine the remaining resistors.

4. (Optional) To further ease the task of obtaining resistor values, set both input resistors to 10 kilohms as well. Our experience is that for the filters we are designing, this has little effect on either the dynamic range or the overall gain.

The method outlined here efficiently produces broadband designs. The result of applying this procedure to the unfiltered response shown in Fig. 2 is shown in Fig. 4. The amplitude and phase response of this filter is shown in Figs. 5a and 5b. Using optional step 4 above, the overall gain of the highpass section is 6.73 and that of the lowpass section is 3.73. The resistance values, in kilohms, are:

Highpass	Lowpass
R1 = 10.0	R1 = 10.0
R2 = 63.7	R2 = 16.3
R3 = 58.3	R3 = 10.0
R4 = 10.0	R4 = 37.3

A single FORTRAN program was written to perform the design steps, menu driven and producing graphical display of the response as the design proceeds. A listing is available for the cost of photocopying, and the source is available provided a 9-track tape is sent to the authors.

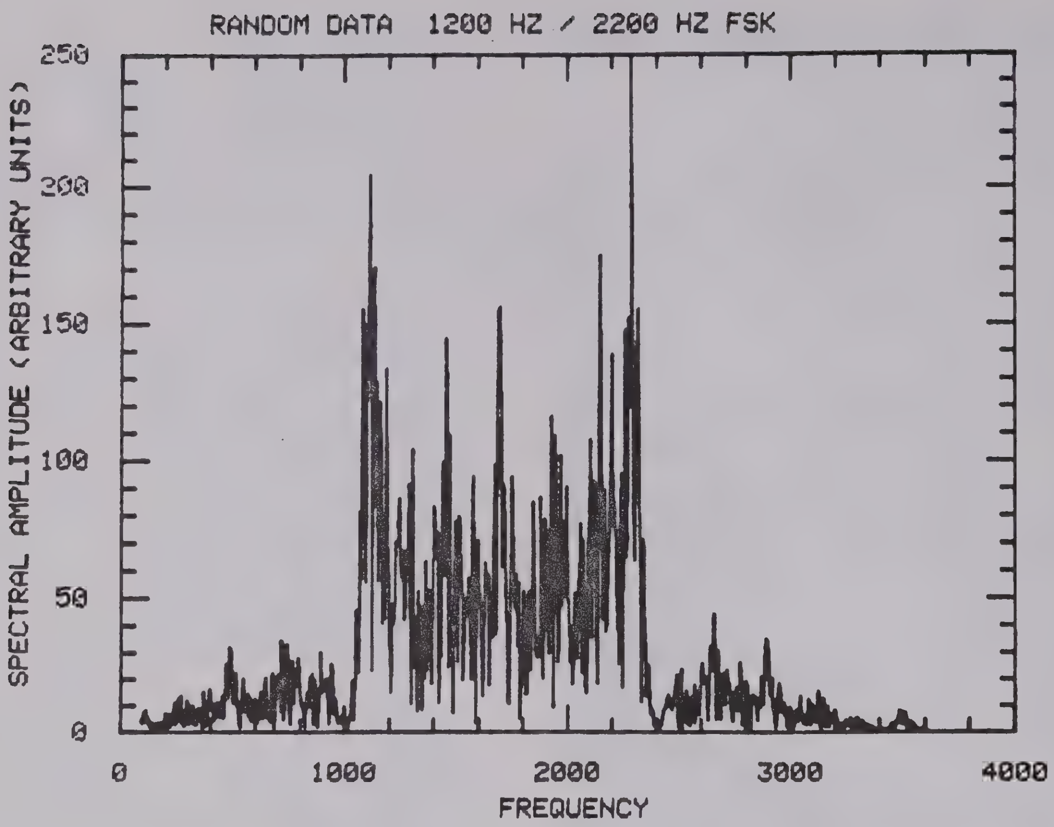


Figure 1. Typical spectral characteristics of random Bell 202 compatible phase-continuous FSK data.

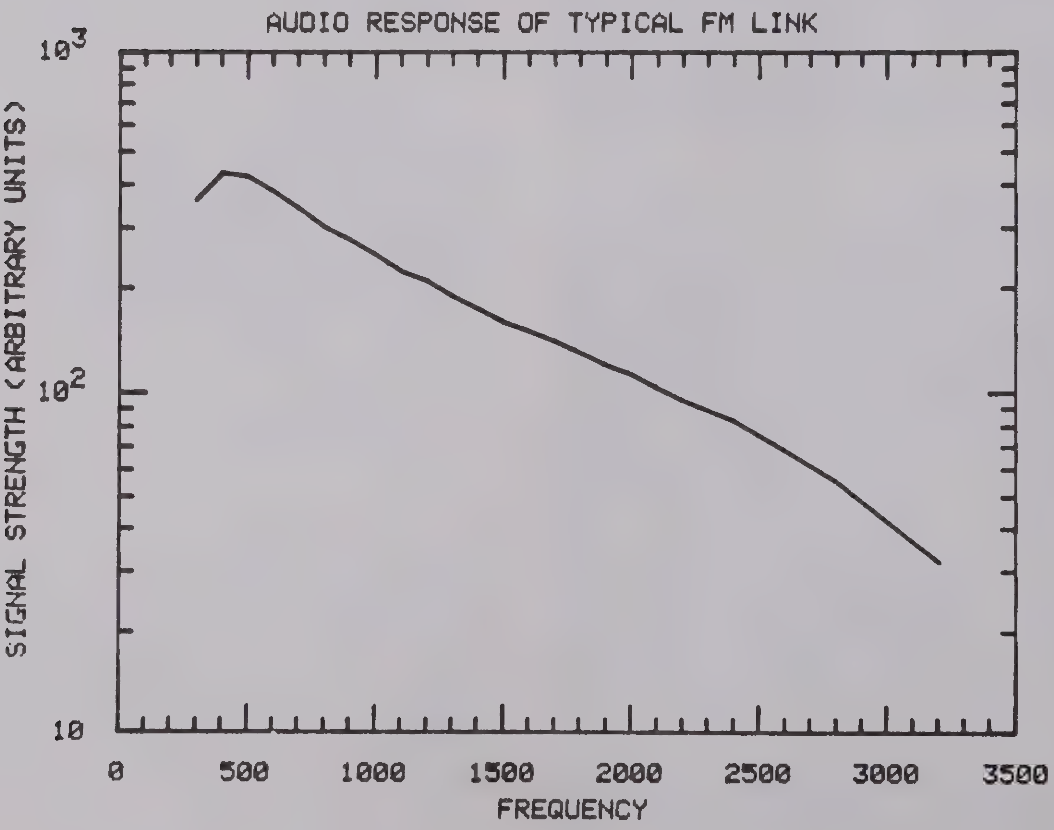


Figure 2. Audio response of a typical 2-meter FM link.

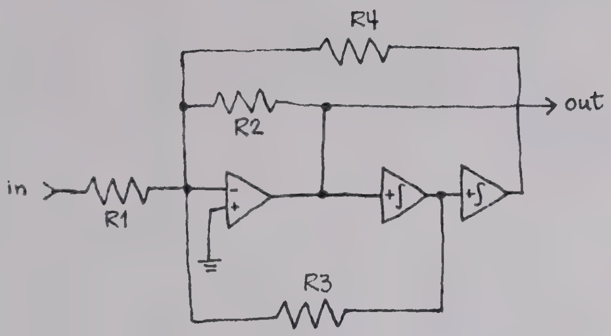


Figure 3a. Equivalent circuit for the input highpass filter section.

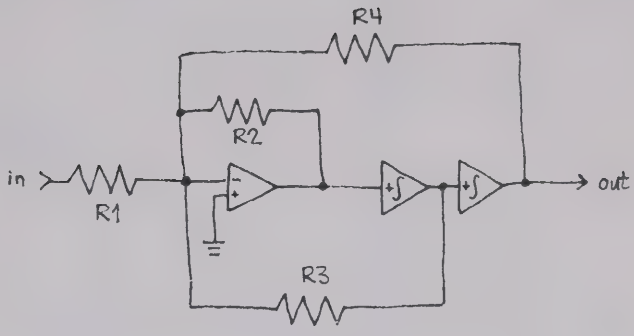


Figure 3b. Equivalent circuit for the output lowpass filter section.

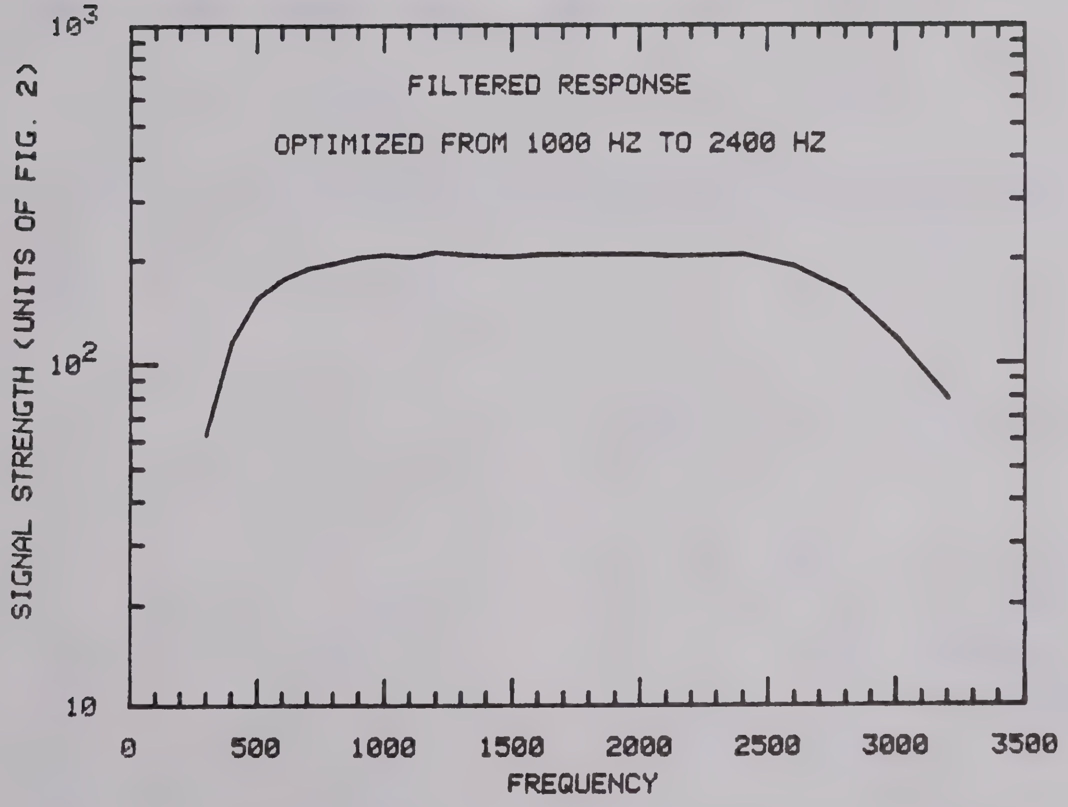


Figure 4. Filtered audio response of Fig. 2

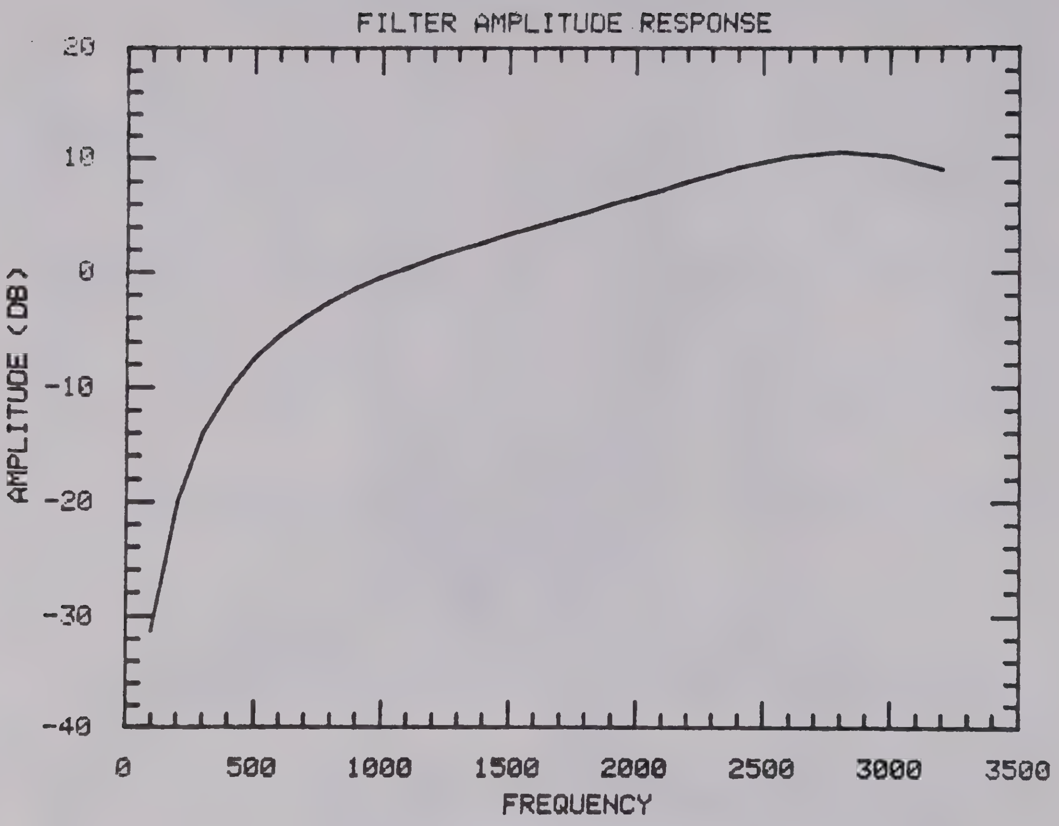


Figure 5a. Filter amplitude response.

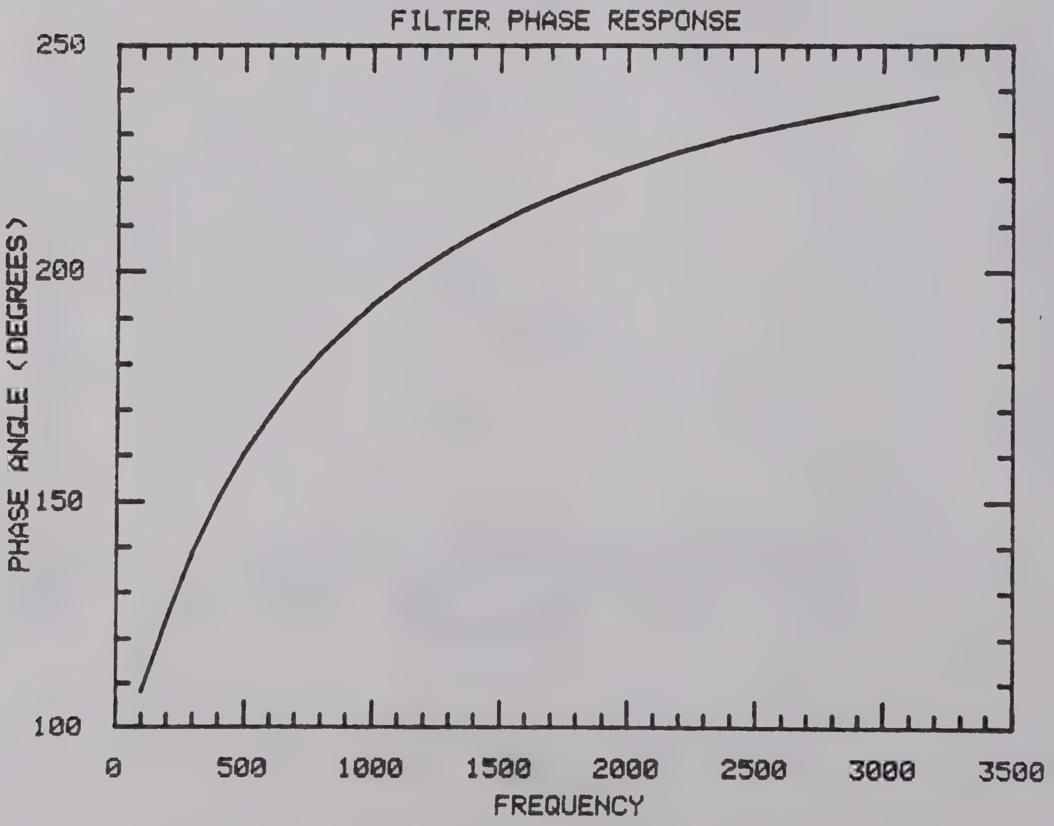


Figure 5b. Filter phase response.

PACKET RADIO FOR EMERGENCY COMMUNICATIONS

Bob Neben, K9BL
126 E. Schantz Ave.
Dayton, Ohio 45409

There is a need to redesign the techniques we use to handle emergency traffic. Many of us are combining processor controlled equipment and traffic handling techniques designed in the 1930's.

Traffic handling originated in radio, using CW, a continuance from the landline systems, I presume. This limits our copy to about 15 to 25 words per minute, depending upon the operator's ability. The reliability of this system is very good since a CW signal can punch its way through a lot of QRM and QRN. Accuracy, however, is limited to the accuracy of the sending operator and the receiving operator, both of whom are subject to fatique.

SSB or FM adds a new dimension, though, and we can talk about 150 to 200 words per minute. At these speeds, however, QRM is more of a problem. Also, we cannot pass traffic at that speed. Assuming we have to write the traffic on a message form, our speed decreases to about 25 words per minute, and we are really not much more ahead of the process than we were with CW. I remember copying MARS (Military Affiliate Radio System) traffic, and whenever possible, checking the addressee in the telephone book. It seemed more often than not that at least one digit was wrong.

RTTY automates what we were doing manually at speeds of 60 to 100 words per minute. Reliability is about the same as voice, and accuracy is slightly better. Maintaining good accuracy requires careful tuning, listening for a "hit", and human attention while typing. Generally I felt the accuracy of our MARS traffic left a lot to be desired.

The type of traffic influences both speed and accuracy. Ragchewing requires neither speed, accuracy, or hardcopy. "Traffic", such as health, welfare, or greeting messages is different. Any media or system we use has a maximum capacity. For instance, suppose we are passing messages using 100 words per minute RTTY, with no QRM, by continuously feeding paper tape to our TD (Transmitter Distributor). The system capacity would approach 100 words per minute in this case, and accuracy of the system would be very good. The dashed line (Figure 1) represents our system capacity. If we are using 60 words per minute RTTY, voice or CW, the dashed line would represent a different system capacity. Equally important though, is the type of traffic.

Normal day-to-day message traffic such as MARS, demands only a small percentage of system capacity. Even at peak periods such as holiday traffic, it can normally be handled during the alloted time for the traffic net. In Figure 1, traffic supplied equals traffic demanded, which is still below system capacity. System accuracy is fairly good since there is time for retransmission requests and no one is under any particular pressure.

Special events such as weather nets or public service events are difficult, as the traffic is not constant. System capacity is still constraining us, and the traffic demanded begins, reaches a peak, and tapers off, (Figure 2). In the case of a weather watch, there is a scramble to get the watchers in position. Traffic builds as the NWS (National Weather Service) , EOC (Emergency Operating Center), or whomever we are assisting demands more information. Usually, just about the time information is most critical, such as when the storm is directly overhead, the system becomes overloaded, and traffic demands have exceeded capacity. What happens? Well, if the net control can keep a cool head and the net is well disciplined, some of the more routine traffic becomes delayed. Accuracy decreases, however, and sorting priorities becomes a problem. Is the Mayor's "routine" acted upon before the NWS "priority"? In time the delayed traffic is transmitted, but some of it will disappear, because it is no longer timely. This is not to imply it wasn't important, it was important, but we missed our chance. Somehow we need a better way of conducting traffic nets.

Disaster nets have a terrible efficiency, (Figure 3). The traffic demands build to gargantuan proportions following tornado touchdowns, and other major events. We work our system to capacity, but it takes days and even weeks to chip away at the workload. Accuracy is horrible, and faith in the system and amateur radio suffers in the long run. I could justify this scenario in the 1930's, but what do we answer in the computer age?

The answer to this problem is to move that system capacity line up so high that we couldn't run into it if we tried and at the same time do error checking to insure 100% system accuracy. This is exactly whatPacket Radio will do for us in the amateur community, and it will do this at a relatively low cost.

A Packet Radio station consists of your present rig (1930 vintage if you so desire, but preferably a modern FM transceiver), some kind of terminal or personal computer, and a TNC (terminal node controller), which does the packet formatting, error checking, and several other functions. Computers are becoming available for $100.00 and up, and TNCs, such as the one offered by the Tuscon Amateur Pack Radio group (TAPR), sell in the $200.00 range. So the cost to upgrade your station to Packet Radio is perhaps the cost of a two meter rig.

Packet Radio will do a number of things for us. It will change the system capacity line from 100 words per minute in our example (74 Baud) to 1200 Baud. On paper that's a sixteen fold increase. In reality, it will be less because of packet overhead, but the increase is still phenomenal. The accuracy is virtually 100%, because of error checking and system acknowledgements. Previously, the net controls could only talk to one station at a time. In Packet Radio, the 64 stations in each local area network (LAN) can send data to other stations simultaneously.

Computers don't have much effect on our present traffic systems, since human intervention is usually required to check status, stored messages, etc. In Packet Radio, we have many uses for the computer. We could store messages for a station not yet logged in. We could do inquiries, such as health and welfare traffic. This may best be done computer to computer, which is fairly easy to set up. We could tie our computer to others in the area over land line, or another frequency to handle incoming traffic. The possible uses for our home and club computers is seemingly endless.

Our traffic nets are usually single function; VHF for local, HF for large areas, etc. By using the gateway function, our LAN Packet System can access worldwide via satellite. This provides a means to get traffic in and out of the local system. Perhaps we need local stations to handle the LAN. The other four stations (or more) could link to other LAN's, gateways, computers, etc.

What could happen if the national emergency evacuation plan were implemented? Imagine moving 100,000 people in your community to an area 50 miles away. It is logical that amateur radio would be used to help coordinate this massive effort. How would we handle this? The logistics would involve massive vehicle movement, fuel, food, medical care, etc. Our present system would have little usefulness, but a Packet Radio system could easily accommodate this. If one LAN becomes overloaded, just initiate another. The gateways would also be heavily used and again, if a gateway becomes overloaded even at 19.4 K baud, another gateway would be initiated.

We are still using old technological equipment. Old technological traffic handling techniques are effective for day-to-day operation, but become overloaded at the first sign of large scale activity. We have the technology to correct the situation, but we need to act now to adapt Packet Radio technology and procedures to our traffic handling system.

Chart

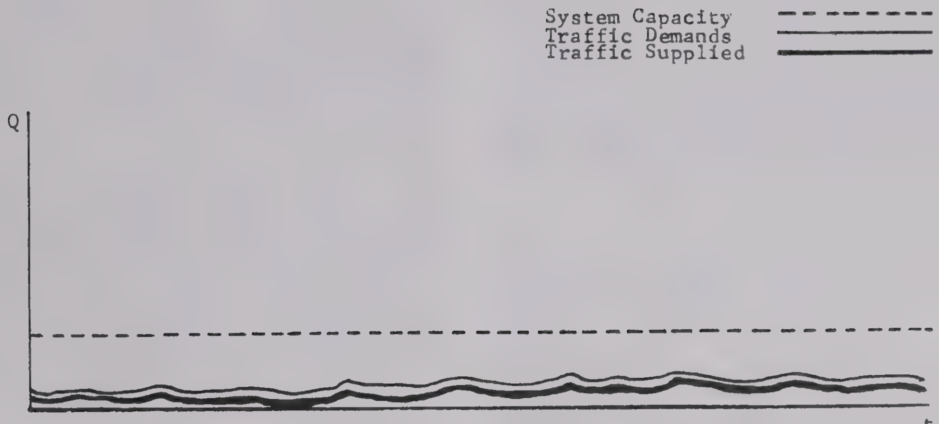


Figure 1 Traffic handling during non-emergencies

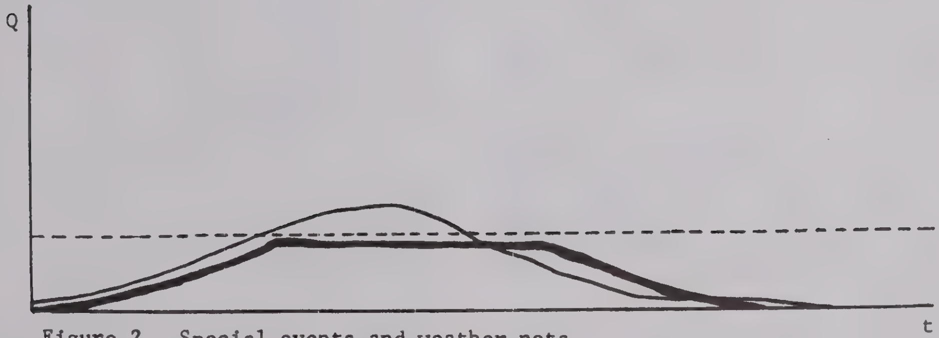


Figure 2 Special events and weather nets

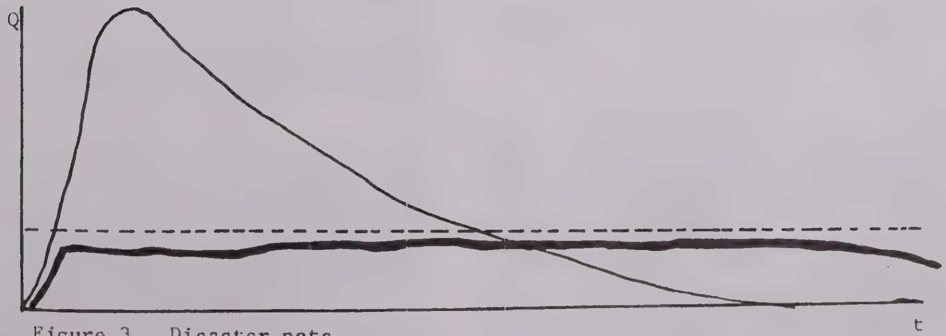


Figure 3 Disaster nets

Traffic Activity Chart

MULTI-USE DESIGN CONSIDERATIONS FOR THE TAPR TNC

Harold E. Price, NK6K
2110 Farrell Ave #14
Redondo Beach, CA 90278

Abstract

The Amateur Packet Radio Terminal Node Controller (TNC) built by TAPR Inc. is designed to meet the needs of a wide range of users, from the technical experimenter to the "appliance operator" end user. This paper discusses the human engineering factors which went into the design of the TNC's external software interfaces that enable it to serve a heterogeneous user base.

Background

The first Terminal Node Controller widely distributed for amateur use was produced by the Vancouver Area Digital Communications Group. It was mainly distributed in bare-board form, and aside from the specialized HDLC communications chip, was able to be stuffed with readily available components. Once the user supplied a modem and power supply he had a device which required only a terminal and a radio to get on the air and send packets. The capability supplied by the VADCG TNC to newcomers in the field of packet radio far surpassed anything else available at the time, and has earned it its place in the Amateur Radio Hall of Fame.

The TAPR TNC was designed to go beyond the capabilities offered by the previous TNCs. Taking advantage of advances in technology, including the reduction in price of denser memories, the TNC hardware offered the following features:

- 24K ROM
- Non-volatile RAM
- 6K RAM
- Onboard power supply with an off board transformer included.
- Onboard modem
- The TNC would be offered assembled and tested.

The increased program space made possible increased software functionality. The increased level of pre-packaging would make the board attractive to an additional class of users, the "appliance operators". This term applies to that group of people who buy a product and expect to plug it in and use it. The term is sometimes used in a derogatory sense but that is not the intent here. This class of TNC user is not interested in the underlying concepts and does not wish to tinker with the innards of the TNC. They accept packet radio as a proven technology and want to get on with the business of being end users, setting up higher level networks, establishing mailbox systems, building local area nets with intelligent host nodes and file servers, or simply using the TNC as error free RTTY. The TNC is a secondary item, simply a means to an end.

This type of user would not be happy if he was required to modify source code and recompile to change node address, call signs, and other operational parameters. On the other end of the spectrum is the experimenter, those individuals interested in tinkering with the lower levels. Many areas of amateur packet radio remain open issues. What are the best timing algorithms? What are the correct retry counts, timeout values, keyup delays, packet lengths and other details associated with the delivery of HDLC frames from one TNC to another? The functions offered by the TAPR TNC to solve these basic compatibility problems are discussed in the following sections.

Conflicting Goals

To illustrate the types of problems involved, the following items are presented, each with its set of conflicting viewpoints.

Degree of Transparency

In applications where the TNC is treated as a means for connecting two devices, computer to computer or remote terminal to computer, the TNC should be totally invisible. It can be viewed as a simple modem, receiving characters from data terminal equipment (DTE) and sending it down a phone line, and receiving data from the line and sending it to the DTE. The TNC has no personality of its own.

In other applications, the TNC is itself the smart device at the end of a communications link, using a terminal only as an IO device under its control. The TNC performs local input editing and output formatting. It attempts to preserve visual fidelity on the terminal device, i.e., not mixing the echo of data being entered for transmission with data received from the link.

Command set

Infrequent (casual) users require a simple command set with straightforward syntax and easy-to-remember english-like commands. Changing TNC major operating modes should require a small number (one) of commands.

Expert users or those experimenting with operational parameters require a rich command set. This class of user also prefers commands needing only a small number of keystrokes as the command entry rate is higher.

Tuning Knobs

This is closely related to command set, in that the more things there are to adjust, the more commands will be required to adjust them.

To use a radio analogy, some hams are happy with ON/OFF, push to talk, and frequency tuning, others require variable bandwidth, IF shift (both high side and low side), notch, RF attenuation, multiple tuning rates, ten VFOs and snooze alarm.

In addition, experimenters want access to all parameters which control forming, sending, and receiving packets.

These are examples of the type of conflicting interests generated by a diverse user base. Availability of a 24K program space on the TNC coupled with the means to develop software in a high level language (Pascal) permitted the three-person software team to provide a single integrated software package which meets everyone's needs. The major features of this design are discussed below.

Definition of Terms

Several of the shorthand terms used in the remainder of this paper are defined here.

User interface. The asynchronous data path to the TNC, the path used to exchange data which is formatted and transmitted over the air. Configuration commands are also given to the TNC through this path.

Link interface. The bit stream (HDLC) path which connects directly to the RF transceiver. Packets (frames) are sent over this path.

Connected. A TNC is "connected" when it has exchanged the proper sequence of packets over the link interface with another TNC such that the two TNCs are connected at the AX.25 (or VADCG) level 2 layer. Data sent by the TNC will be in the form of sequenced (numbered) information frames. Connected does not refer to any physical connections.

Unconnected. The TNC is not "connected" to any other TNC. Data sent by it on the link is in the form of unsequenced information frames.

Data transfer mode. Characters received through the user interface are assumed to be data for eventual transmission through the link when the TNC is in a data transfer mode. When not in this mode data input is assumed to be a configuration command.

Operating modes

The major point of difference between users of the TNC is the degree of transparency of the user interface when the TNC is in a data transfer mode. Put more simply, is the TNC something one speaks TO or speaks THROUGH? As stated previously, some users want to treat the TNC as a smart terminal, others view it as a smart modem, to be heard but not seen. Because of the large number of differences in basic TNC functions required to satisfy these two needs, the TNC software supplies two data transfer modes. Entering one of these modes has the effect of changing the values of a large number of configuration parameters at once. The data transfer modes are the CONVERSATIONAL mode and the TRANSPARENT mode, and are abbreviated to "convers" and "trans".

Changing modes causes a radical change in behavior of the interface. Attributes of the TNC in CONVERS mode are:

1) Characters input are scanned for special meaning to implement features discussed below.

2) Creation of packets is directly under user control. A special character is used to say "take the data entered previously, make it into a packet, and send it as soon as possible".

3) Data can be edited until released to be made into a packet. Characters can be struck out, a line erased, or the entire packet can be erased.

4) Data input is echoed locally, i.e., by the TNC.

5) Data received from the link is formatted before being output.

6) Flow control is available via use of XON/XOFF characters. Flow control is the process where data buffer or CRT display overflow is avoided by stopping the flow of data. The XOFF character is a request to halt the flow of data, XON is permission to resume the flow through the user interface. Case translation is performed, line feeds can be inserted in the data stream following carriage returns, carriage returns are inserted when the output line length is exceeded.

The attributes of the TNC when in the TRANS mode are:

1) All characters are transferred to the link without modification.

2) Creation of packets is only under the indirect control of the user. Packets are formed by time or by data length. Data length control causes packets to be formed after "n" characters have been entered through the user interface. Time control will build a packet every "n" seconds or after "n" seconds have passed since the last character was received from the user interface.

3) Local echoing and local editing are not available.

4) Data received through the link is transferred to the user interface as received, without modification or addition of control characters.

5) Flow control is not controlled by data characters, but is managed by use of the standard RS-232 handshake lines.

These two data transfer modes allow the TNC to be compatible with a wide range of applications. The behaviors described are present by default, they may be modified at will by the user. Functions can be added to TRANS mode, or removed from CONVERS mode, supplying a variety of hybrid semi-transparent or translucent modes.

A third operating mode is present in the TNC, COMMAND mode. Command mode is active when the TNC is first powered up, and can be entered from either of the other modes. COMMAND mode is used to change the configuration and operating parameters of the TNC. Changing the TNC from mode to mode does not effect the state of the link, i.e., whether or not the TNC is connected. This allows two personal computer owners to talk back and forth in CONVERS mode, and then enter TRANS mode to let their computers transfer files.

A problem arises with respect to the total transparent mode. Short of a hardware reset, how is the command mode entered from the transparent mode? The method used to leave convers mode and enter command mode, a special control character, can not be used. Instead, a small compromise is made. After a user defined period (guard time) has passed where no characters have been received from the user interface the TNC will watch for a mode escape character. If this character is entered three times with no intervening characters, and another guard interval passes with no additional input, the TNC enters the command mode. A proper selection of the guard time and escape character will provide a high probability that reception of the sequence is a valid request for command mode entry. This procedure did not originate with the TAPR TNC and is a common practice for intelligent modems.

Parameters

There are many facets of the user and link interfaces that experimenter will want to twiddle with and the end user will want to ignore, or at most change only once. Examples of user interface items are:

- o Editing characters such as character delete, line delete, and packet delete.
- o Flow control characters.
- o Packet creation times in TRANS mode.
- o Output presentation control such as <lf> after <cr>, number of nulls after <cr>, and terminal line length.

Examples of link control items are:

- o Transmitter and repeater keyup delays.
- o Number of times to retry a frame, frame acknowledge time
- o Packet data size
- o Maximum contiguous frames sent or maximum number of outstanding unacknowledged frames
- o Station address or call sign.

Previous amateur packet systems kept these parameters in ROM and supplied no way to modify them at run-time. The TAPR TNC software embodies a design methodology where all such values are kept in RAM while the TNC is running, and therefore can be modified by user command. The values are initialized from ROM when the TNC is first installed. The large number of parameters available to the user (see table 1) would require a very tedious startup procedure every time power was applied as not all users would be happy with the default values supplied in ROM. To avoid this unpleasantness, the TAPR TNC uses non-volatile RAM to store parameters when the TNC is turned off. A dip switch on the TNC selects either the ROM copy or the non-volatile RAM copy for initialization of the RAM parameters. After an initial boot from ROM, the TNC is then configured to use the non-volatile RAM for subsequent booting.

This combination of software and hardware design permits hands-off operation for some users, one-time configuration for end-users, and endless opportunity for change by experimenters.

Commands

The desire to have many user accessible parameters conflicts with the desire to have a small number of simple commands. Since a major design goal was to externalize as many operational parameters as possible, it was soon realized that the TNC would have several commands. The Beta test release of the TNC software has 66 separate commands.

Only four commands, the commands to connect and disconnect the link and the mode change commands, are used in normal operation of the TNC. Most of the other commands are used to change parameters which have default values pre-assigned when the TNC is first powered on. These values are satisfactory for most applications and will never be changed. If the parameters are changed, however, the user can issue a command that causes the changed values to be stored in the non-volatile RAM, replacing the default values.

The command syntax is kept simple, consisting only of the name of the parameter to be changed and the new value. A DISPLAY command is present which lists all parameters and their values. While not a replacement for a HELP facility, the list of parameters names serves as a memory jogger and will cut down on trips through the manual.

To reduce the number of keystrokes needed, all commands can be abbreviated to a smaller number of characters that still uniquely identify the parameter.

Documentation

As much care was taken with the design and implementation of the manual as was taken with the design and implementation of the hardware and software. The documentation must serve the same diverse user base the that TNC does. Because of the need to provide both tutorial and detailed information, the manual for the TNC is approximately 140 pages long. The manual is structured to supply the right level of detail at the right time. The first few pages deal directly with the tasks required to get the TNC on the air, including a detailed radio interface section. This detail is required because incorrect wiring of the TNC to RF interface can result in damage to the TNC and/or the RF gear. The introductory material is keep as short as possible however, and by page 28 the user will be on the air sending packets.

Subsequent sections of the manual provide tutorial information on the hardware and on the protocols used. Other sections provide detailed information on the software command set and on the TNC hardware. A full set of schematics is provided for hardware experimenters. Also included in appendicies are the specifications for the communications protocols used on the TNC. These supply sufficient detail for others to implement compatible software.

The manual is structured to supply the level of detail required by experimenters while at the same time not scaring off the casual user.

Summary

Great care was taken during the design and implementation of the TNC software to provide a simple interface for end users while not limiting the activities of more advanced users. The TAPR TNC is currently in a Beta Test phase with 180 users spread out in more than 16 local groups. It is expected that new commands will be needed while some current commands will prove to be superfluous and can be removed. The large percentage of high level code and human engineered design should make this a relatively painless task.

Acknowledgements

Many thanks go to Margret Morrison, KV7D, and Dave Henderson, KD4NL, who along with the author designed and implemented the TAPR AX.25 software, and to Dan Morrison, KV7B, for his design inputs.

Command	Type	Function
ABAUD	U	Asynchronous baud rate
ABIT	U	Asynchronous stop bits
AUTOLF	U	Follow received <cr> with <lf>
AWLEN	U	Asynchronous word length
AX25	L	AX.25 or VADCG protocol
AXDELAY	L	Keyup delay time for audio repeater
AXHANG	L	Audio repeater dropout (hang) time
BEACON	L	Beacon mode
BKONDEL	U	Echo <bs> when <del> entered
BTEXT	L	Beacon text
CALIBRAT	C	Enter hardware calibration routine
CANLINE	U	Cancel line character
CANPAC	U	Cancel packet character
CMDTIME	U	Command mode entry from TRANS mode timer
COMMAND	U	Command entry from CONVERS mode character
CONMODE	D	Default data transfer mode
CONNECT	L	Connect TNC to another TNC
CONOK	L	Permits connections in unattended operation
CONVERS	D	Enter CONVERSation mode
CPACTIME	U	Create packets by time in CONVERS mode
CR	U	Append a <cr> to the end of each packet in CONVERS mode
DEBUG	C	Enter the debugger
DELETE	U	Specifies which of <del> or <bs> deletes characters
DIGIPEAT	L	Enables AX.25 digipeating function
DISC	L	Disconnects the TNC from the link
DISPLAY	C	Displays all configuration parameters
DWAIT	L	Digipeater wait time
ECHO	U	Local echo in not in transparent mode
ESCAPE	U	Character echoed when <esc> is entered
FLOW	U	I/O data flow in CONVERS mode
FRACK	L	Frame acknowledge timeout
HBAUD	L	HDLC (link) baud rate
ID	L	Send CW id
LCOK	U	Lower case translation
MAXFRAME	L	Maximum unacknowledged frames
MONALL	L	Link monitor command
MONCON	L	Link monitor command
MONFROM	L	Link monitor command
MONITOR	L	Link monitor command
MONTO	L	Link monitor command
MYCALL	L	Call sign (address) of this TNC in AX.25 mode
MYVADR	L	VADCG protocol address byte
NULLS	U	Number of nulls added after <cr> in CONVERS mode
PACLEN	L	Maximum packet data length
PACTIME	U	Packet creation time for TRANS mode
PARITY	U	Parity of asynchronous user interface
PASS	U	Send next character verbatim in CONVERS mode
PERM	C	Put current parameter setting in non-volatile RAM.
RESET	C	Software reset
RETRY	L	Frame retry count
SCREENL	U	Screen width for CONVERS auto-wrap feature
SENDPAC	U	Create packet character for CONVERS mode
START	U	Flow control character
STOP	U	Flow control character
TRACE	L	Link diagnostic command
TRANS	D	Enter TRANSparent mode
TXDELAY	L	Transmitter keyup delay
TXFLOW	U	Use character flow control in TRANS mode
UNPROTO	L	Unconnected address field
VDIGIPEA	L	Enables VADCG digipeater function
VRPT	L	Direct VADCG frames sen by this TNC to a digipeater
XFLOW	U	XON/OFF or hardware flow control in CONVERS mode
XMITOK	L	Enable transmit functions
XOFF	U	Flow control character
XON	U	Flow control character

L - Link commands	U - User interface
C - TNC control	D - Data transfer mode

Table 1. TNC commands

Packet Radio - A Software Approach

Robert M. Richardson, W4UCH
22 North Lake Drive
Chautauqua Lake, N.Y. 14722

Abstract:

A software rather than hardware approach to synchronous Packet Radio communication at 1200 or 2400 Baud using the Radio Shack TRS-80, Model I or Model III microcomputer is described. The program duplicates virtually all the functions provided by the Vancouver Area Digital Communications Group (VADCG) terminal node controller board which requires an 8085 microprocessor, an 8273 synchronous data link controller (SDLC), an 8250 serial I/O, and a number of EPROM and RAM memory chips, plus a separate microcomputer with RS232C interface and a 1200/2200 Hz modem.

The only external equipment required by the software approach, other than a TRS-80, amateur VHF transceiver and antenna, is a port zero encoder/decoder, and two EXAR chips for AFSK keying and demodulation. The program has been extensively tested on the 2 meter amateur band working into southern Ontario and locally in western New York.

Introduction:

Packet radio communications is coming down the amateur radio pike in the near future, and the near future is now upon us. We have closely followed the evolution of amateur packet radio with its asynchronous beginnings in the Montreal area during early 1979 and its synchronous beginnings in the Vancouver area later in 1979/1980. The synchronous packet radio pioneers, including Douglas Lockhart VE7APU et al in the Vancover area developed the rightly famous and widely used VADCG terminal node controller board which was the introduction to synchronous amateur packet radio for the majority of all amateurs actively participating in packet communications today.

Though a number of SDLC controllers in addition to the Intel 8273 are now available, such as those from Western Digital and Zilog, and the price will hopefully be coming down as volume increases, there is yet another approach to amateur packet radio which due to its low cost and simplicity, may greatly broaden amateur participation in this wave of the future. This approach is the software rather than firmware approach using a Model I or Model III TRS-80 with 48K memory and hopefully for convenience, 1 or 2 mini-disk drives. Suffice it to say, assembly language is used which offers nearly 300 times faster execution speed than standard Fortran, Pascal, or Basic high level languages.

Even with the remarkable speed assembly language offers the user, it does require a finite amount of time in the receive mode for the program to serially accomplish what the dedicated SDLC chips are able to accomplish in parallel; i.e., find the last openning flag and address and store the packet bits in memory, convert the serial packet bits to decimal with zero deletion (the opposite of zero insertion), and do the CRC checking for each frame. Nevertheless, this entire process for the average packet only requires 90 milliseconds which is not a significant or even noticeable time delay to the operator at either end of the circuit.

General:

The program is comprised of 5 segments:

1. The transmit mode segment which does the work horse job of converting prepared messages or programs, either keyboard or disk input, into IBM SDLC format, and clocking them out bit by bit serially at the desired Baud rate via port zero. Packet and frame length may be any length desired from 1 to 250 bytes plus address, control and CRC bytes, and may be input from the menu. See figure 1. A total packet or frame length of 256 bytes will be allowed when intra repeater routing comes to pass. The extra two bytes will serve to determine routing. The number of preamble flags sent before the packet may also be programmed which is a courtesy for those with slow transmit and receive switching at the receiving end. A number of prepared messages may selected from the menu as well as keyboard input to memory for a nearly immediate packet. The video display utilizes the split screen format with the top 8 lines for receive and the bottom 8 lines for transmit. See figure 2. Independent sequential scrolling is provided for both transmit and receive modes.

2. The cyclic redundancy check,

CRC16, segment which automatically generates the 2 byte CRC in IBM modulo 2 format and appends these 2 bytes to each frame transmitted. This same subroutine serves the receive mode to check each incoming frame for CRC validity also using the IBM modulo 2 CRC algorithm which the dedicated SDLC chips utilize.

3. The receive mode segment has a software equivalent of a digital phase locked loop which takes the serial packet input via port zero, and stores each bit in memory either 8 bits per byte for best memory utilization, or 1 bit per byte for instructional purposes (easy to visualize). After the complete packet has been received, it is first converted from binary to decimal, then stored in high memory and displayed on video while the CRC test is being made for each frame if a multi frame packet was received. Depending upon whether the packet is of the unnumbered, supervisory, or information variety, appropriate action is taken automatically. The conventions followed are those used by the VADCG so the program is fully compatible with those amateurs using the Vancouver terminal node controller. The program will receive and decode packets of any length up to the 12,288 byte capacity of the bit store allocation. Average packet decoding time after reception of the packet transmmission is 90 milliseconds, as previously mentioned.

4. The edit/modify segment is a unique and extremely useful utility that allows the operator instant access to any 1 of the 1024 byte, 60 pages of memory in the TRS-80. Each page of memory is displayed at one time on the video display in TRS-80 ASCII or graphics format. Either upper case only, or upper and lower case edit/modify may be selected from the menu. Pressing the 'M' for modify key intiates a flashing cursor that may be moved up, down, left, or right on the displayed page with the arrow keys. Keyboard input ASCII value may be input directly if desired or pressing the shift '@' keys displays the memory location in decimal, memory value at this location, stack pointer value, and the operator asked to input the new value. Upon inputting the new value, the full memory page is displayed again. Pages are moved up or down through memory by pressing the BREAK key (flashing cursor disappears) and then the ENTER key to move up a page in memory or the minus/DASH key to move down a page in memory. Needless to say, though ROM may be examined with this function, only RAM may be modified. To eliminate tedious paging by hand, a number of control keys allows instant access to the more frequently used memory locations; i.e., transmit program store, received bit store, and received message store.

5. Morse code I.D. and Morse transmit segment. This minor subroutine called by shift 'I' sends the Morse I.D. of the transmitting station at any speed desired to satisfy F.C.C. requirements. Conversely, Morse code at any speed may be tranmitted from the keyboard via shift 'X'.

Memory Management:

The operating program resides in memory from 29696 through 40959. Three 12,288 byte segments are used exclusively by:

A. 17408-29695 is reserved for the program or data to be transmitted store. Normally, a disk program to be transmitted via packet is first loaded from disk to 40960+ in memory and then moved down to the 17408+ area by pressing shift 'Y'. In the 'connected' mode of operation the program or data from 17408+ is automatically transmitted in single frame packets of 254 bytes length (a western New York convention so that a single station may NOT monopolize the packet repeater unfairly). This automatic transmission is called by pressing 'B' from the menu. Acknowledgement (ACK) packets are received automatically and if valid the next packet automatically transmitted, otherwise the previous packet is retransmitted.

B. 40960-53247 is reserved for incoming received bit storage. We prefer to store a bit per byte so that the user may visualize the stored bit pattern, though 8 bits per byte could be used for better memory utilization. Unless directed by the menu command to SAVE bit storage, this area is automatically cleared after each packet is converted to decimal and stored in high memory. Surely all you amateurs operating VADCG terminal node controllers know that it sends two SDLC logical zero 'bytes' AFTER the opening flags, which are then followed by a single flag BEFORE the address byte. Or did you? Packets less than 4 bytes total length, are ignored by the SDLC protocol that is used in this program.

C. 53248-65535 is utilized for storing the converted decimal byte values from received packets. When it is full, an automatic 'not ready to receive' (RNR) = 'wait' is transmitted and when acknowledged, the operator may clear out all address, control, and CRC bytes by pressing '$' from the menu. Pressing shift 'B' then takes the operator to DOS ready where the received program or data may be DUMPed to disk, after which one returns to the program, sends a 'ready to receive' (RR) to 'clear' the previous 'wait' and continues upward and onward if it is indeed a program or data longer than 12,288 bytes bytes in length.

Conclusion:

Amateur radio has room for all types and varieties of individuals with dramatically differing interests. Some prefer operating and could care less how an electron gets from 'A' location to 'B' location and what is involved in getting it there. There is certainly nothing wrong with that, and to that variety of amateur we highly recommend the Vancouver terminal node controller kit *. With a few hours of soldering you are 'on the air' using their EPROMs. To the variety of amateur who truly wishes to thoroughly understand how the absolutely brilliant IBM SDLC protocol operates, how to modify it for the forthcoming HDLC modes, we recommend the software rather than hardware approach. When you are done, you will not only be an 'instant SDLC expert,' but a better assembly language programmer too.

The software approach to packet radio communications is obviously limited by the clock speed of the microcomputer being used. The Model I and Model III TRS-80 will easily handle 1200 and 2400 Baud packets, with 4800 Baud packets being somewhat marginal but acceptable, without modifying the standard crystal clock of either microcomputer. By installing one of the numerous clock speed up kits available (4 MHz), both 4800 and 9600 Baud packets could be handled.

We are indebted to a number of our Canadian neighbors in the Hamilton and Toronto area for assistance in testing the software approach to packet radio. Most noteable have been VE3MWM, VE3DSP, VE3IUV, and VE3DVV. Their packet repeater located in the southern environs of Toronto with the apt call sign of VE3RPT is now active on 145.650 MHz.

Locally (65 miles northeast of our QTH), the major effort to install a packet repeater in the greater Bufffalo area has been borne by W2EUP with considerable assistance from VE3MWM. It will be linked to both the Toronto repeater to the north and a new packet repeater in the Syracuse area that will be able to link into the greater New York City area to the east, and thence linked up and down the east coast.

We have only touched briefly on a few of the highlights of the software approach to packet radio communications. To go through it in depth would take much more than the allotted space and time. For those who wish to explore the subject in greater depth, at least a few hundred pages worth, we suggest you watch for the new book, 'The Packet Radio Handbook' that will be published spring '83 by Richcraft Engineering Ltd., #1 Wahmeda Industrial Park, Chautauqua, N.Y. 14722 at $22 postpaid to the U.S. and Canada and $30 postpaid overseas.

* Vancouver Area Digital Comm. Group
818 Rondeau Street
Coquitlam, British Columbia
Canada V3J 5Z3

```
                  ENTER OPTION DESIRED ? _

CHANGE ADDRESSEE & NUMBER   = A    W2EUP  CONNECT REQUEST CQ   = B
NOT CONNECTED TOGGLE        = C    W2EUP  DISCONNECT REQUEST   = D
SEND PACKETS FROM LO-MEM    = E    W2EUP  CONNECT ACKNOWLEDGE  = F
USING ONLY SOFTWARE MSG     = G    WORKING ON AMSAT AX.25 MSG  = H
NOT INSERT DPLL BIT TOGGLE  = I    SEND HI-MEM CONTINUOUSLY    = J
NOW IN UPPER CASE MODIFY    = K    FILL HI-MEM WITH UUUUU      = L
DISPLAY/EDIT MEMORY PAGE    = M    SET DISCRETE PACKET LENGTH  = N
LOAD HI-MEM ALL 11111       = O    CHANGE DPLL TIMING VALUE    = P
SEND CONTINUOUS FLAGS/126   = Q    ABORT LO-MEM XMIT SEQUENCE  = R
MULTI-FRAME PACKET TEST     = S    QUICK BROWN FOX TEST MSG    = T
CLEAR NON-PGM MEM 17K-65K   = U    INPUT/TRANSMIT MESSAGE = V & W
ANY PACKETEERS ABOOT 255/0  = X    MOVE MID-MEM TO LO-MEM      = Y
NORM BIT STASH CLEAR        = 1    MOVE RECV PACKS TO LO-MEM   = 2
SET NUMBER OPENING FLAGS    = 3    WAIT & CLEAR WAIT TOGGLE    = 4
```

Figure 1

```
1200 BAUD SDLC RECEIVE MODE                ----> NOW CONNECTED
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1200 BAUD SDLC TRANSMIT MODE               CONNECTED TO VE3MWM
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 2

INTRODUCING THE PACKET ADAPTIVE MODEM (PAM)

Paul L. Rinaldo, W4RI
Amateur Radio Research and Development Corp. (AMRAD)
P.O. Drawer 6128
McLean, Virginia USA 22106

Abstract

This paper describes a modem design undertaken by Robert E. Watson and the author. The modem was designed primarily for high-frequency packet radio applications. It operates at signaling rates of 75, 150, 300, 600 and 1200 bauds. The data rate is software controllable through a modified RS-232-C port. A frequency shift of 600 Hz is maintained for all data rates. The modulator is phase continuous and provides X32 or X64 clock to the packet assembler/disassembler (PAD) or terminal-node controller (TNC). The demodulator employs a National MF10 switched-capacitor filter (SCF) chip for each of the 1500-Hz mark and 2100-Hz space frequencies. Bandwidths of the MF10s are software controllable to accommodate different received data rates and receiver frequency tolerances. A point-to-point wired prototype of the modem has been built on an S-100 perf board. Power may be taken from the S-100 bus or provided by a separate power supply. The prototype has been laboratory tested with excellent "eye diagrams" on speeds up to 600 baud with some eye closing at 1200 bauds. Still pending is a design decision whether to combine an optimal minimum-shift keying (msk) demodulator circuitry for 1200-baud operation with this modem or to make it a separate modem. Upon completion pc boards and documentation will be made available to amateurs.

Baudot Radioteletype - Some Background

High-frequency (hf) radioteletype (RTTY) using frequency-shift keying (fsk) began in the U.S. Amateur Radio Service in 1953 when the Federal Communications Commission (FCC) authorized F1 emission in the hf bands. Virtually all teletypewriters at that time were military or commercial surplus. They used a version of a five-unit code (U.S. Military Standard or CCITT International Telegraphic Alphabet No. 2) usually referred to as the Baudot or Murray code. Five-unit teletypewriters used since the 1950s in the U.S. normally operated at a speed of 45.45 baud (60.61 wpm), but a few 56.92-baud (76.68-wpm) and 74.2-baud (100-wpm) machines were also used. Machines available from European sources ran 50 baud (66.67 wpm).

Many fsk demodulators (also called "tuning units" or TUs) were home-brewed by amateurs. However, the design standards were those of military and commercial RTTY demodulators. In the 1950s, the demodulators were normally designed to receive two audio frequencies separated by some multiple of 170 Hz, usually 850 Hz. The favorite frequencies for many years were 2125 (mark) and 2975 Hz (space). In time, amateurs experimented with narrower shifts, particularly 170 Hz, using the audio frequencies 2125 (mark) and 2295 Hz (space), and standardized on it in the late 1970s. Today, virtually all amateur Baudot RTTY demodulators use 170-Hz shift, although many also accommodate 850-Hz and 425-Hz shifts as well. Many hf RTTY stations use transceivers in the ssb mode, sending afsk into the microphone input of the transmitter and obtaining afsk output from the receiver audio stages. Many of these transceivers start rolling their audio off not much above 2000 Hz. As a result, the afsk frequencies 1275 Hz (mark) and 1445 (space) (called the low tones) are also used and are standard in Europe.

The majority of these afsk RTTY demodulators were designed as fm demodulators. In this type of demodulator, the signal is first sent through a bandpass filter to remove out-of-band interference and noise. It is then limited to remove amplitude variations. The signal is fm-demodulated in a discriminator or a phase-locked loop (PLL). The output of the detector is run through a low-pass filter to remove noise at frequencies above the baud rate. The result is fed to circuit which makes the decision between binary 1s and 0s.

ASCII Radioteletype

Effective March 17, 1980 FCC rules authorized the use of the American Standard Code for Information Interchange (ASCII) as defined in American National Standard Institute (ANSI) Standard X3.4-1968.

Straightforward asynchronous serial ASCII has not been very popular on the ham bands since it was legalized. This is due to a variety of reasons. Some RTTYers own mechanical teletypewriters, mostly 45-baud Baudot. They currently are being phased out in favor of electronic digital terminals or computers which can communicate in either Baudot or ASCII. AMTOR is now added to the list of modes possible with "glass TTYs."

Probably the reason why ASCII has not caught on in the hf bands is that some amateurs have experienced poor results trying to operate 300-baud ASCII. The lack of success is largely due to the modem design limitations.

In order to operate existing Baudot demodulators at the higher signaling rates used in ASCII, it is necessary to raise the cut-off frequency of the low-pass filter, and it may be necessary to redesign the bandpass filter and any filters used in the detector. The design can usually be stretched to copy 110 bauds. However, poor results can be expected when trying to modify most 170-Hz, 45-baud demodulators to receive signaling rates above 110 bauds.

Other problems inlcude intersymbol distortion due to multipath propagation. Multipath can be reduced or eliminated by operating near the maximum-usable frequency (muf). These problems are discussed in my paper given at the first packet conference.[RIN81]

Amateur ARQ and FEC Hf RTTY Systems

Although hf skywave RTTY is difficult, high-quality error-free operation has been achieved by two robust systems using automatic repeat request (ARQ) and forward error control (FEC). One is AMTOR.[MAR81], [FCC RM-4122], [MEI82] Another experimental system was designed by Jerome Dijak, W9JD.[DIJ81-83]

Amateur Hf Packet Experiments

Here is a summary of U.S. amateur hf packet experimental contacts:

On February 8, 1982 K1RT in Connecticut and W9LLO in California made a brief connection over 20 meters using Collins KWM-380s, Vancouver TNCs and hf RTTY modems.

On May 31, 1982, K8MMO and W4RI, both in Northern Virginia, carried on a two-hour connection on 10 meters, at 1200 bauds, using ICOM IC-701's, Vancouver TNCs and Bell 202 modems.

On October 16, 1982, a 15-minute connection took place between W3IWI in Maryland and and N5AHD in Texas using Vancouver TNCs and Bell 202 modems.

Design Considerations

Data Rates

For this modem design, we are primarily concerned with signaling speeds which are feasible for passing through unmodified audio sections of Amateur Radio transceivers, particularly hf-ssb transceivers.

The FCC rules permit up to 300 bauds on frequencies between 3.5 and 21.25 MHz, up to 1200 bauds between 28 and 50 MHz, 19.6 kilobauds between 50 and 220 and 56 kilobauds above 220 MHz.

In addition to the regulatory restriction, the upper signaling rate may be limited on hf skywave due to intersymbol distortion introduced by multipath propagation. This and related subjects were analyzed in a paper I presented at the first packet conference in 1981.[RIN81] For some general reading on the behavior of the hf skywave medium for data communications see [BRA75].

Speeds up to 1200 bauds should be practical on the hf amateur bands whenever a usable frequency exists near the maximum usable frequency (muf). At present, use of a 1200-baud signaling rate below 28 MHz requires a Special Temporary Authority (STA) from the FCC. I plan to apply for an STA in the near future. I also plan to investigate the feasibility of a permanent rules change to permit up to 1200 bauds on all hf bands, including a portion of the 160-meter band where F1 emission is not now permitted.

At the lower end of the speed range, some consideration was given to including the rate of 37.5 bauds for those infrequent occasions when 75 bauds can't be made to work due to intersymbol distortion. The speeds of 18.75 and 9.375 bauds could have been included but were rejected as being too slow.

We decided to make the speeds 75, 150, 300, 600 and 1200 bauds. These five speeds are selected by means of on-board solid-state switches. The modulator clock output rate is controlled by a 4512 8-channel buffered data selector. Filters in the demodulator are set by six 4051 single 8-channel analog multiplexer/demultiplexers. These seven chips are controlled by three lines from the PAD.

Using five speeds and having an 8-channel switching capability permitted the inclusion of wider bandwidths for the three slower speeds of 75, 150 and 300 bauds. The wider-bandwidth capability is to make allowance for frequency error. This is particularly useful when the receiver's frequency is digitally controlled and/or left unattended. The ICOM IC-720A is subject to a frequency error of ±50 Hz when externally controlled. On the other hand, a no-compromise narrower bandwidth can be selected when the receiver is front-panel controlled, thus settable to within ±5 Hz.

Frequency Shift

The 170-Hz shift in common use for Baudot RTTY could be used for data rates of 75 and 150 bauds with signal-to-noise (S/N) ratios common on amateur hf RTTY. Use of this shift at 300 bauds has been done with some sacrifice of demodulator error performance. It is not suitable for the speeds of 600 and 1200 bauds.

When the narrower shifts (say below 400 Hz) are used, there is a tendency for the mark and space frequencies to fade dependently (together). So, if one fades, the other is likely to fade at the same time. The mark and space frequencies tend to fade more independently when the shift is wider. Independent fading is common at the age-old shifts of 425 and 850 Hz, with more independence obseved for 850 Hz shift. Some commercial RTTY demodulators make use of this so-called in-band frequency diversity and use combining and/or selection techniques to continue copying even if one frequency or the other fades completely.

We have chosen a shift of 600 Hz for two reasons. One is that there is enough frequency separation to permit good in-band frequency diversity action. Also, the frequency of 600 Hz is directly related to the baud rates and permits phase-continuous modulation and synchronous demodulation. At 1200 bauds, a 600-Hz shift is called minimum-shift keying (msk) or sometimes fast frequency-shift (ffsk) to connote that the shift is less than 1 Hz per baud.

Physical Construction

We decided to use an (IEEE 696) S-100 card for the modem. This 5- x 10-inch board was about the size needed for all the chips if a double-sided printed wiring is used. The modem can be plugged into an S-100 computer frame and take power from the bus. Or it can be mounted in its own box with a separate power supply if desired. None of the S-100 data or control lines is used.

Having tasted the fruits of receiving RTTY with a polarization-diversity setup, I plan to build a second PAM board for the second demodulator channel. One demod will be fed by an IC-720A transceiver, the other by an IC-R70 receiver. I also just purchased an IC-7072 interface unit which slaves the two together. The first PAM will have the modulator and a demodulator. The second board is to have an identical demodulator (same pc pattern) and, in the place of the modulator, a diversity selector to process the outputs of the two demodulators.

Input/Output Connections

The (data) I/O connection to the PAD follows EIA RS-232-C rules with the exception that the three data-rate control lines (pins 18, 23 and 25) are presently at TTL levels to reduce the PAD chip count. Insulation-displacement connector (IDC) headers are used on both the PAD and PAM boards rather than the bulkier DB-25. If there is need to route this I/O outside the cabinet, a cable with an IDC plug would connect to a back-panel-mounted DB-25.

The (analog) I/O connection to the radio is to be done with an IDC header. Work remains to be done on the radio side of the modem to ensure compatibility with various amateur hf radios. The goal is to devise an interface scheme that will permit the greatest flexibility.

The interfacing of the transmited analog (TxA) and received analog (RxA) signals is only a matter of adjusting levels and keeping unwanted rf at arm's length. However, the IC-720A and the IC-R70 (and some other ICOM radios) have a 24-pin accessory connector which permits digital remote control of frequency. As some external control circuitry is needed for the ICOMs, one of my future projects will be to design a Receiver Interface Board (RIB) to go between the modem and the radio(s) when ICOMs are used. The RIB will also be built on an S-100 card. As the RIB will be optional, the connector scheme will be designed to work with the RIB or without it.

The I/O design has been deferred until receipt of a Tuscon Amateur Packet Radio (TAPR) beta test model TNC. Although designed as a companion modem for the AMRAD PAD, the goal is to make the PAM usable with the preexisting Vancouver and TAPR TNCs before the I/O design is finalized.

Circuit Description

Fig. 1 shows the shematic of the PAM as of this writing. CMOS logic integrated circuits are used throughout.

Modulator Circuitry

A 2.688-MHz crystal master clock feeds two divider chains. This crystal is not on a stock frequency and was ordered from a crystal manufacturer.

The U2-U6 chain divides by 32, then divides by 7 for mark or 5 for space. U4 divides by 8 and feeds its outputs to two U5 Ex-OR gates to produce a stepwise approximation of a sine wave at either 1500 Hz (mark) or 2100 Hz (space). The U6 active filter removes the steps to produce a nearly sinusoidal output to modulate the transmitter.

The other divider chain produces either X64 or X32 clock for the PAD or TNC respectively for the signaling rates of 75, 150, 300, 600 and 1200 bauds. The U9 binary counter delivers a number of outputs which correspond to X64 clock at each rate. These outputs are selected by U10. A divide-by-2 counter of U4 is used for an X32-clock output.

Demodulator Circuitry

A commonly stocked 2.097152-MHz crystal was used as the master clock to 49.94 times the center frequencies of the two MF10 filters. This stock crystal (and divider chain) just happens to clock the MF10 filters for center frequencies of 1499.8 and 2099.7 Hz.

U109 is the MF10 space filter, U110 the mark filter. Both sections of each MF10 are used to achieve a 4th-order bandpass filter with steep skirts to cope with hf band crowding. Four 4051 8-channel analog multiplexers, U105-U108, select resistance values which set filter bandwidths.

The first halves of U111 and U112 are active low-pass filters for the mark and space tones and provide feedback to the MF10s to reduce tilt. The other halves are full-wave detectors.

The first half of U113 sums the detected outputs and does some low-pass roll off. The main post-detection low-pass filtering is accomplished in the second half of U113 with U114 and U115 there to switch resistor values according to baud rate. U116's two halves are positive and negative peak detectors.

One half of U117 is a comparator for the positive and negative levels and feeds the RxD to the PAD via part of U5 which can be used to invert data. Inverting data is not necessary when NRZI encoding is used but may be needed for NRZ-encoded signals.

The other half of U117 is a tuning indicator circuit. An on-board LED is switched in for testing and out when an external LED is connected. Here's how to use it: While tuning the receiver, control the input level (RxA) for less than half brightness. Tune for maximum brightness, reducing the receiver gain as necessary.Increase the input signal level until a maximum is reached, then back it off to half brightness.

Of course, an oscilloscope would provide a better tuning display. One can be connected at points X and Y fed by the first halves of U111 and U112.

Semiconductors

Q1	2N2222	npn transistor
U1,101	CD4049	hex inverting buffer
U2,9,102	CD4024	7-bit binary counter
U3,7,8,103,104	CD4029	presettable binary/decade up/down counter
U4	CD4520	dual binary counter
U5	CD4077	quad ex-OR
U6	LM741CN	op amp
U10	CD4512	8-channel data selector
U11	MC1489	RS-232-C receiver
U12	MC1488	RS-232-C driver
U105-108,114,115	CD4051	single 8-channel analog multiplexer/demultiplexer
U107,108	MF10BN	universal monolithic dual switched capacitor filter
U111-113	MC1458	op amp
U116,117	TL082	op amp

Acknowledgements

This is to thank Bob Watson for his design work and many patient hours discussing approaches and to acknowledge contributions to the design made by Terry Fox, WB4JFI.

References

[BRA75] Kenneth Brayer, "Data Communications via Fading Channels," New York, NY: IEEE Press, 1975.

[DIJ81-83] Jerome Dijak, W9JD, "The W9JD HF ARQ/FEC System," AMRAD Newsletter series, August, October, November and December 1981, January, February, March, April, November and December 1982, and January 1983.

[EIA RS-232-C] "Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange," Electronic Instries Association, August 1969.

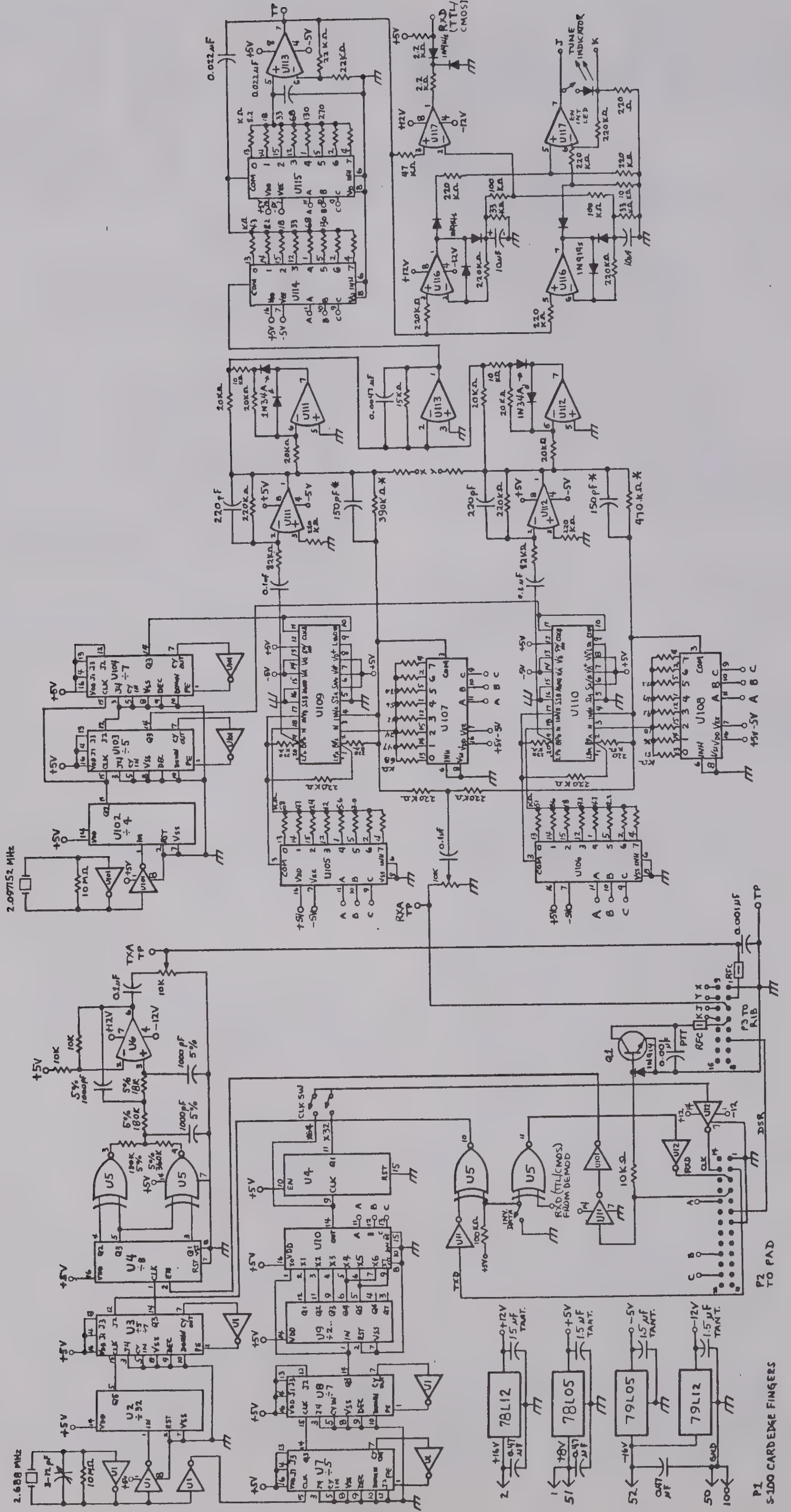
[FCC RM-4122] "Authorization of the digital code "AMTOR" for use by stations in the Amateur Radio Service," PR FCC 83-36 32687, released February 8, 1983.

[MAR81] J. P. Martinez, G3PLX, "Amtor, an Improved Error-Free RTTY System," QST, June 1981.

[MEY82] William Meyn, K4PA, seminar on AMTOR, Teleconference Radio Net, December 2, 1982.

[RIN81] Rinaldo, Paul L. Rinaldo, W4RI, "Amateur Packet Network Agenda," proc. ARRL Amateur Radio Computer Networking Conference, October, 1981.

[WAT80] Robert E. Watson, "FSK: Signals and Demodulation," Watkins-Johnson Company Tech-notes, September/October 1980.



2.688 MHz
2.097152 MHz
U2 ÷32
U3 ÷7
U4 ÷8
U7 ÷5
U8 ÷7
U9 ÷2
U10
U102 ÷4
U103 ÷5
U104 ÷7
U105
U106
U107
U108
U109
U110
U111
U112
U113
U114
U115
U116
U117
TXA TP
RXA TP
RXD (TTL/CMOS)
TUNE INDICATOR
CLK SW
X32
RXD (TTL/CMOS) FROM DEMOD
TXD
DSR
PTT
P3 TO RIB
P2 TO PAD
78L12
78L05
79L05
79L12
1.5 µF TANT.
P1 S-100 CARDEDGE FINGERS

SOFTNET - AN APPROACH TO HIGH LEVEL PACKET COMMUNICATION

Jens Zander and Robert Forchheimer
Dept. of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden

Abstract

SOFTNET is a packet-radio concept under development in Sweden. The network is distributed and all nodes are programmable via the network during normal operation. This concept represents an unconventional approach to the protocol issue and offers elegant solutions to the higher level communication problems. This paper gives a programming model of the network, along with some illustrating examples.

I. Introduction

The SOFTNET approach was conceived in 1980 and discussed among Swedish radio amateurs. The discussion led to a proposal for an experimental network in the 432 MHz band utilizing bit rates up to 100 kbps. During 1981 this draft was presented to the Swedish Telecommunication Administration. The Administration responded in a positive way, giving the packet radio group at Linköping University virtually free hands. This group, consisting of 6 people, is currently involved in developing prototype nodes and basic software for the network.

The main concept behind SOFTNET is that all packets are considered to be programs of a network language. These programs are interpreted in the nodes as soon as they arrive. Nodes can be programmed by any number of users simultaneously without unwanted interaction. This approach makes it possible for a user to define his own high level services like datagrams, virtual calls, file transfers and mailboxes. The concept also allows changes at lower levels during operation, permitting redefinition of LINK-level/Access protocols. A detailed description of these ideas can be found in [1], [2], [3], [5].

II. Node model

In a SOFTNET node, an incoming packet that has passed the link level is given to the node computer for interpretation. Here, a standardized set of instructions are available. The kernel of this set is simply a FORTH interpreter to which has been added functions that control the node hardware. Thus, any user may execute his own FORTH program in any of the nodes that he can reach. This way he is able to instruct another node to either deliver the packet to the owner of that node or to retransmit it so that the node merely acts as a repeater. FORTH allows the creation of private directories so the user may also store programs in remote nodes. These programs may either wake up upon the arrival of a packet from the user or upon an internal signal (e.g. the real time clock) produced by the remote node itself. Describing the node thus reduces to describing a programming model. In the FORTH case, this is done by simply listing all the available functions or "words". [4]. Fig. 1 summarizes the packet format from the user's point of view.

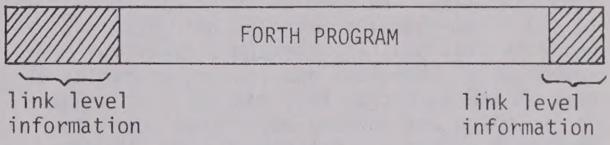


Fig. 1. A SOFTNET packet

In fact, the link level protocol has been added to the FORTH kernel so that also the link level information is handled by a FORTH interpreter. This permits on line reprogramming and extension of the link protocol such as new version of HDLC, access algorithms etc. Thus, from the first byte to the last, a SOFTNET packet is simply a set of FORTH statements. From a practical point of view it is a good idea to conceptually keep instructions at the link level apart from the higher level programs since changes at link level have to be coordinated among the users.

III. The Node - a multiuser/multitasking system

Processing at the link level requires real time performance while higher level tasks are less time constrained. On the other hand, the link processor serves one packet at a time sequentially while higher level tasks may run concurrently. Also, the programming activities of one user should not influence any other. Thus, a SOFTNET node must be able to support parallel tasks besides being able to keep apart the current users of the node. For the prototype implementation our choice was a dual processor (6809) system. One of the processors are solely devoted to link level processing. The second processor contains a multitasking FORTH interpreter and is shared among the users. A special task - the owner process - interfaces the node to the owner's equipment which can be anything from a dumb terminal to a fullgrown computer system. In the latter case the dual processor FORTH system is simply considered a modem between the owner's system and the network.

IV. Node programming example

Consider the simple network given in fig. 2. Here four nodes are connected by two-way radio paths as indicated by the arcs between the nodes.

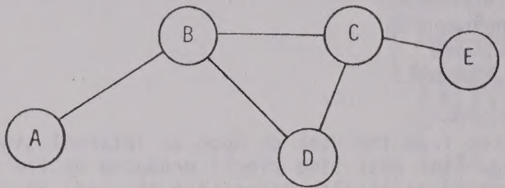


Fig. 2. Sample Network

Suppose a user is located at node A and has the specific task to deliver a large number of packets to node E, i.e. he wants to establish a "virtual" call to E. This can be done in at least two ways. The simplest thing to do is just to add a retransmit command to all packets as is shown in fig. 3a). The command % takes the next symbol as an (one-hop) address and transmits the rest of the packet to that address. This goes on, dropping one address each time, until the remaining packet reaches node E were the data portion is transferred to the OWNER of the node. This procedure may however consume valuable packet space, especially when many intermediate nodes are used. We can instead make use of the programming

```
% B   % C   % E   OWNER  <data>          a)


% B   : VCE ." VCE" % C ;

                                         b)

% B   % C   : VCE ." OWNER" % E ;

: VCE  ." VCE" % B ;

VCE  <data>                              c)
```

Fig. 3. Node programming

feature and instruct the intermediate nodes B and C just to pass along the packet to the next node in line. This can be done as in fig 3b). Here we define a new function, named, say, VCE in the intermediate nodes and our own node. A new definition is made FORTH-style starting with a ":" and ending with a ";". The effect of executing VCE is to pass on the rest of the packet to the next node in line. Also, the function places a copy of its name first in the packet for repeated execution in the succeeding nodes.

V. Project status

Since the advent of the project at Linköping University, a rapidly growing number of interested radio amateurs have joined the discussions. A SOFTNET user group, SUG, is being formed as a subgroup of AMSAT-SM. Up to date this group has received about 100 applications for membership.

Hardware development has made considerable progress. The Node-computer board is under production and a first shipment of 50 kits was delivered in February. Also the PC-layout for the LINK-computer board is completed. The packet-radio utilizes a duobinary direct FSK modulation scheme with favourable bandwidth properties. Transmission is synchronous and MFM coding is used to recover clock information. Due to problems in the design of the radio testing of the digital hardware and software had to be done on a cable bound local network. A system with up to 4 nodes has so far been successfully demontrated and has provided useful results for further software development.

VI. Conclusions

The SOFTNET concept with its fully programmable nodes will give the user opportunity not only to communicate, but to conduct experiments in network architecture and network protocols. The concept is applicable to all kinds of communication networks. An implementation using a local network cable has been successfully tested and a UHF-radio broadcast network is under construction [2].

VII. References

[1] Persson, I., Forchheimer, R.: Design Considerations of a Distributed Packet Radio Network using the Amateur Radio bands, Internal Report, LiTH-ISY-I-0408, May 1980.

[2] Zander, J., Forchheimer, R.: Preliminary Specifications for a Distributed Packet Radio Network for Computer- and Radio Amateurs, Internal Report, LiTH-ISY-I-0424, January 1980.

[3] Zander, J., Forchheimer, R.: Softnet-Packet radio in Sweden ARRL Amateur Radio Computer Networking, Conf. Gaithesburg, MD, October 1981.

[4] Forchheimer, R., Zander, J.: Softnet - User's Manual, Linköping 1983.

[5] Qvigstad, F., Matts, S.: Construction of a packet radio Node computer, Internal Report, LiTH-ISY-I-0491, December 1981.